

MMM	MMM	TTTTTTTTTTTTTTTT	AAAAAAAAA	AAAAAAAAA	CCCCCCCCCCCC	PPPPPPPPPPPP	
MMM	MMM	TTTTTTTTTTTTTTTT	AAAAAAAAA	AAAAAAAAA	CCCCCCCCCCCC	PPPPPPPPPPPP	
MMM	MMM	TTTTTTTTTTTTTTTT	AAAAAAAAA	AAAAAAAAA	CCCCCCCCCCCC	PPPPPPPPPPPP	
MMMMMM	MMMMMM	TTT	AAA	AAA	CCC	PPP	PPP
MMMMMM	MMMMMM	TTT	AAA	AAA	CCC	PPP	PPP
MMMMMM	MMMMMM	TTT	AAA	AAA	CCC	PPP	PPP
MMM	MMM	TTT	AAA	AAA	CCC	PPP	PPP
MMM	MMM	TTT	AAA	AAA	CCC	PPP	PPP
MMM	MMM	TTT	AAA	AAA	CCC	PPP	PPP
MMM	MMM	TTT	AAA	AAA	CCC	PPP	PPP
MMM	MMM	TTT	AAA	AAA	CCC	PPPPPPPPPPPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPPPPPPPPPPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPPPPPPPPPPP	
MMM	MMM	TTT	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	CCC	PPP	
MMM	MMM	TTT	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	CCC	PPP	
MMM	MMM	TTT	AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCCCCCCCCCCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCCCCCCCCCCC	PPP	
MMM	MMM	TTT	AAA	AAA	CCCCCCCCCCCC	PPP	

MOU
VO4

```

LL                      IIIIII                      SSSSSSSS
LL                      IIIIII                      SSSSSSSS
LL                      II                      SS
LL                      II                      SS
LL                      II                      SS
LL                      II                      SS
LL                      II                      SSSSSS
LL                      II                      SSSSSS
LL                      II                      SS
LL                      II                      SS
LL                      II                      SS
LL                      II                      SS
LLLLLLLLLLLLLL        IIIIII                      SSSSSSSS
LLLLLLLLLLLLLL        IIIIII                      SSSSSSSS

```


K 9
16-Sep-1984 02:25:33
14-Sep-1984 12:46:44

VAX-11 Bliss-32 V4.0-742
[MTAACP.SRC]MOUVOL.B32;1

Page 1
(1)

MOU
V04

```
0001 0
0002 0 MODULE MOUVOL (LANGUAGE (BLISS32) ,
0003 0 IDENT = 'V04-000'
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *
0028 1 *****
0029 1
0030 1 ++
0031 1
0032 1 FACILITY: MTAACP
0033 1
0034 1 ABSTRACT:
0035 1 This module mounts a volume
0036 1
0037 1
0038 1 ENVIRONMENT:
0039 1
0040 1 VMS operating system, including privileged system services
0041 1 and internal exec routines.
0042 1
0043 1 --
0044 1
0045 1
0046 1
0047 1 AUTHOR: D. H. GILLESPIE, CREATION DATE: 24-AUG-1977
0048 1
0049 1 MODIFIED BY:
0050 1
0051 1 V03-013 HH0041 Hai Huang 24-Jul-1984
0052 1 Remove REQUIRE 'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.
0053 1
0054 1 V03-012 MMD0288 Meg Dumont, 10-Apr-1984 14:14
0055 1 Fix to the return from $MTACCESS code where ACCESS could
0056 1 be set to normal processing before all the error conditions
0057 1 where checked.
```

58	0058	1	
59	0059	1	
60	0060	1	
61	0061	1	
62	0062	1	
63	0063	1	
64	0064	1	
65	0065	1	
66	0066	1	
67	0067	1	
68	0068	1	
69	0069	1	
70	0070	1	
71	0071	1	
72	0072	1	
73	0073	1	
74	0074	1	
75	0075	1	
76	0076	1	
77	0077	1	
78	0078	1	
79	0079	1	
80	0080	1	
81	0081	1	
82	0082	1	
83	0083	1	
84	0084	1	
85	0085	1	
86	0086	1	
87	0087	1	
88	0088	1	
89	0089	1	
90	0090	1	
91	0091	1	
92	0092	1	
93	0093	1	
94	0094	1	
95	0095	1	
96	0096	1	
97	0097	1	
98	0098	1	
99	0099	1	
100	0100	1	
101	0101	1	
102	0102	1	
103	0103	1	
104	0104	1	
105	0105	1	
106	0106	1	
107	0107	1	
108	0108	1	
109	0109	1	
110	0110	1	
111	0111	1	
112	0112	1	
113	0113	1	
114	0114	1	

V03-011	LMP0221	L. Mark Pilant,	28-Mar-1984 14:45
	Change UCBSL_OWNUIC to ORBSL_OWNER and UCBSW_VPROT to ORBSW_PROT.		
V03-010	MMD0271	Meg Dumont,	23-Mar-1984 9:34
	Change the processing of the accessibility character fields in the VOL1 label to call the installation specific accessibility routine. The return from this routine determines the users access to the volume.		
V03-009	MMD0185	Meg Dumont,	6-Jul-1983 18:32
	Make the default for AVL/AVR the same from the DCL call and from the system service call.		
V03-008	MMD0176	Meg Dumont,	26-May-1983 15:11
	Fix to support new input to IOC\$CVT_DEVNAM		
V03-007	MMD0174	Meg Dumont,	9-May-1983 15:16
	Fix to make IO_STATUS consistently defined within module		
V03-006	MMD0164	Meg Dumont,	26-Apr-1983 9:43
	Change the references to 80 to be the symbol ANSI_LBLSZ. Change the reference to 240 to be the symbol SCRATCH_OFFSET.		
V03-005	MMD0134	Meg Dumont,	12-Apr-1983 17:24
	Added support for writing and interrupting the VOL1 OWNER IDENTIFIER field, so that it is no longer treated as a VMS field, strictly. Bugfix to the AVL, AVR code where MOUNT/INIT would not work under all circumstances.		
V03-004	MMD0120	Meg Dumont,	29-Mar-1983 0:44
	Added support for the VOL2 label inside the MTAACP		
V03-003	MMD0103	Meg Dumont,	17-Feb-1983 13:14
	Use GET_DEV_NAME to get the tape units device name. Added the routine GET_DEV_NAME to call the system routine IOC\$CVT_DEVNAM to get the tape units name. Added the code to do automatic volume recognition and labeling (AVR and AVL).		
V03-002	MMD0002	Meg Dumont,	3-Jan-1983 15:43
	Allow user with read access to a tape to mount the tape writelocked. Add modifier IOSM_CLRSEREXCP to all QIO's issued by the MTAACP, necessary for the MSCP tape drives.		
V03-001	MMD0001	Meg Dumont,	23-Mar-1982 10:16
	Added a check for member UIC match when mounting a volume.		
V02-014	DMW00071	David Michael Walp	21-Jan-1981
	Handle Volume Invalid during verification		
V02-014	DMW00059	David Michael Walp	7-Dec-1981
	Moved Rename TRANSLATION_TABLE to ANSI_A_BAD, ANSI_A_GOOD		
V02-013	DMW00036	David Michael Walp	17-Sep-1981


```
115 0115 1 | Correct error messages given ( MOUNT, REMOUNT switched )
116 0116 1 |
117 0117 1 | V02-012 DMW00030 David Michael Walp 18-Aug-1981
118 0118 1 | Volume Access and ANSI 'a' character in Volume Names
119 0119 1 |
120 0120 1 | V02-011 DMW00017 David Michael Walp 20-May-1981
121 0121 1 | Copy the new fields ( File-Set-Id and Vol_Acc ) then
122 0122 1 | creating New MVL.
123 0123 1 |
124 0124 1 | V02-010 DMW00014 David Michael Walp 14-Mar-1981
125 0125 1 | Changed the calculation of the CCB address to GET_CCB
126 0126 1 |
127 0127 1 | V02-009 REFORMAT Maria del C. Nasr 30-Jun-1980
128 0128 1 |
129 0129 1 | A0008 MCN0003 Maria del C. Nasr 15-Oct-1979 9:23
130 0130 1 | Add HDR3 processing
131 0131 1 |
132 0132 1 | **
133 0133 1 |
134 0134 1 | LIBRARY 'SYSS$LIBRARY:LIB.L32';
135 0135 1 |
136 0136 1 | REQUIRE 'SRC$:MTADEF.B32';
137 0520 1 |
138 0521 1 | REQUIRE 'LIBD$: [VMSLIB.OBJ]INITMSG.B32';
139 0653 1 |
140 0654 1 | LINKAGE
141 0655 1 |     LSCHOOSE_UNIT = JSB : GLOBAL (CURRENT_VCB = 11)
142 0656 1 |                     NOTUSED (2, 3, 4, 5, 6, 7, 8, 9, 10),
143 0657 1 |     MVL_UCB = CALL : GLOBAL (MVL_ENTRY = 9, UCB_LIST = 10,
144 0658 1 |                     CURRENT_VCB = 11);
145 0659 1 |
146 0660 1 | FORWARD ROUTINE
147 0661 1 |     ASSUME_MOUNTED : NOVALUE MVL_UCB, | assume correct volume is
148 0662 1 |                     | mounted
149 0663 1 |     CLPREV_MAKECUR : NOVALUE MVL_UCB, | clear prev use of volume and
150 0664 1 |                     | make it current
151 0665 1 |     CHECK_ACCESS : MVL_UCB, | check the access rights to a
152 0666 1 |                     | tape
153 0667 1 |     CHECK_RING : COMMON_CALL, | check for the write ring
154 0668 1 |     CHOOSE_UNIT : LSCHOOSE_UNIT, | choose unit for mount
155 0669 1 |     CLEAR_UNIT : NOVALUE MVL_UCB, | clear last mounted volume
156 0670 1 |     CREATE_LABEL : COMMON_CALL, | create a label for the volume
157 0671 1 |     GET_DEV_NAME : COMMON_CALL NOVALUE, | given the UCB addr get dev nam
158 0672 1 |     MAKE_CUR_VOL : NOVALUE MVL_UCB, | make volume current
159 0673 1 |     MAKE_VOL_ENTRY : COMMON_CALL, | make volume entry in MVL
160 0674 1 |     OPERATOR_LBL : MVL_UCB, | record operator supplied
161 0675 1 |                     | label
162 0676 1 |     SET_MVL_OVERRIDE : NOVALUE MVL_UCB; | set the MVL override bit
163 0677 1 |
164 0678 1 | EXTERNAL
165 0679 1 |     CURRENT_UCB : REF BBLOCK, | address of current unit control block
166 0680 1 |     CURRENT_WCB : REF BBLOCK, | address of current WCB
167 0681 1 |     IO_CHANNEL, | acp io channel
168 0682 1 |     IO_STATUS : VECTOR [2], | status of io
169 0683 1 |     MAIL_CHANNEL,
170 0684 1 |     WORK_AREA;
171 0685 1 |
```

```

: 172 0686 1 EXTERNAL ROUTINE
: 173 0687 1 BLOCK, ! block activity on current volume set
: 174 0688 1 CHECK_PROT, ! check VMS protection rights
: 175 0689 1 ENABLE_MAIL_AST : COMMON_CALL,
: 176 0690 1 GET_CCB, ! get the address of the CCB
: 177 0691 1 ! get devname given its UCB
: 178 0692 1 GET_RECORD, ! get the record tape is currently reading
: 179 0693 1 IOCSVT_DEVNAM : LSIOC CVT_DEVNAM ADDRESSING_MODE (ABSOLUTE),
: 180 0694 1 PRINT_OPR_MSG : L$PRINT_OPR_MSG,
: 181 0695 1 PRINT_NOT_LABEL : JSB, ! print not correct label
: 182 0696 1 PROCESS_VOL2_LABEL, ! interpret the contents of VOL2
: 183 0697 1 READ_BLOCK : COMMON_CALL, ! read mag tape block
: 184 0698 1 REWIND AND WAIT : COMMON_CALL, ! rewind volume and wait for completion
: 185 0699 1 SEND_ERRLOG,
: 186 0700 1 SYSSFAO : ADDRESSING_MODE (ABSOLUTE), ! format ascii output
: 187 0701 1 SYSSQIOW : ADDRESSING_MODE (ABSOLUTE), ! queue I/O and wait
: 188 0702 1 SYSSSETIMR : ADDRESSING_MODE (ABSOLUTE), ! set time request
: 189 0703 1 SYSSWAITFR : ADDRESSING_MODE (ABSOLUTE), ! wait for event
: 190 0704 1 TAPE_OWN_PROT, ! determine the owner and
: 191 0705 1 ! protection of a tape
: 192 0706 1 TERMINATE_VOL; ! terminate mount request
: 193 0707 1
: 194 0708 1 BIND
: 195 0709 1 MAIL = WORK_AREA : BBLOCK [MSGSIZE],
: 196 0710 1 MAILSZ = MAIL + MSGSIZE,
: 197 0711 1 STARID = UPLIT ('DECFILE11A');
: 198 0712 1
: 199 0713 1 OWN
: 200 0714 1 TAPE_OWNER,
: 201 0715 1 TAPE_PROT : BITVECTOR [16],
: 202 0716 1 LABEL_SPEC : BITVECTOR [1], ! set if oper specified label
: 203 0717 1 INFORM_OPER : BITVECTOR [1], ! set if oper should know that
: 204 0718 1 ! a mount happen without their
: 205 0719 1 ! involvement
: 206 0720 1 CVT_DEVNAM : VECTOR [MAX_DEVNAM_LENGTH,BYTE], ! Converted device
: 207 0721 1 ! name
: 208 0722 1 CVT_DEVNAM_LENGTH : BYTE; ! length of device name
```



```
210 0723 1 GLOBAL ROUTINE MOUNT_VOL (VOL, FLAGS) : COMMON_CALL =
211 0724 1
212 0725 1 ++
213 0726 1
214 0727 1 FUNCTIONAL DESCRIPTION:
215 0728 1
216 0729 1 & REWRITE THIS DESCRIPTION
217 0730 1 &
218 0731 1 This routine mounts the specified relative volume. If it is
219 0732 1 already mounted and the rewind flag is set then the volume will be
220 0733 1 rewound and the VOL1 label rechecked. If the volume is not mounted,
221 0734 1 a request is issued to the operator, the process blocks until the oper
222 0735 1 replies that the volume has been mounted. Then if the label flag
223 0736 1 is set, the VOL1 label is checked against the one entered at the
224 0737 1 original mount time or one entered by the operator when indicating
225 0738 1 that the volume was mounted.
226 0739 1
227 0740 1 CALLING SEQUENCE:
228 0741 1 MOUNT_VOL(ARG1,ARG2)
229 0742 1
230 0743 1 INPUT PARAMETERS:
231 0744 1 ARG1 - relative volume number to mount
232 0745 1 ARG2 - flags
233 0746 1 MOUSV_REWIND - request rewind of volume
234 0747 1 MOUSV_LBLCHECK - request check of label
235 0748 1 MOUSV_CHKIFSPC - check the label only if the operator supplied
236 0749 1 MOUSV_MOUNTERR - error on last mount, so force mount
237 0750 1
238 0751 1 IMPLICIT INPUTS:
239 0752 1 NONE
240 0753 1
241 0754 1 OUTPUT PARAMETERS:
242 0755 1 NONE
243 0756 1
244 0757 1 IMPLICIT OUTPUTS:
245 0758 1 volume is mounted and set current
246 0759 1
247 0760 1 ROUTINE VALUE:
248 0761 1 NONE
249 0762 1
250 0763 1 SIDE EFFECTS:
251 0764 1 NONE
252 0765 1
253 0766 1 --
254 0767 1
255 0768 2 BEGIN
256 0769 2
257 0770 2 BIND
258 0771 2 SECONDS = UPLIT (-10000000, -1); : one second in 100 nsec units
259 0772 2
260 0773 2 MAP
261 0774 2 FLAGS : BBLOCK;
262 0775 2
263 0776 2 GLOBAL REGISTER
264 0777 2 MVL_ENTRY = 9 : REF BBLOCK, : addr of MVL entry for current vol
265 0778 2 UCB_LIST = 10 : REF VECTOR; : addr of list of UCB's for vol set
266 0779 2
```



```
267 0780 2 EXTERNAL REGISTER
268 0781 2 COMMON_REG;
269 0782 2
270 0783 2 LOCAL
271 0784 2 MVL : REF BBLOCK; ! address of MVL control block
272 0785 2
273 0786 2
274 0787 2 ! get the MVL and see if we need to increase its size. This means that
275 0788 2 ! if we have more volumes in the set then originally specified then we
276 0789 2 ! must create more MVL enties for those volumes. Each volume in a volume
277 0790 2 ! set has its own MVL for the duration of the mount of that volume set.
278 0791 2
279 0792 2 MVL = .CURRENT_VCB[VCB$L_MVL];
280 0793 2 IF .MVL[MVL$B_NVOLS] LSS .VOL
281 0794 2 THEN
282 0795 2 MVL = KERNEL_CALL(MAKE_VOL_ENTRY, .VOL, .MVL);
283 0796 2
284 0797 2 ! point at the current MVL label
285 0798 2
286 0799 2 MVL_ENTRY = .MVL + MVL$K_FIXLEN + ((.VOL - 1)*MVL$K_LENGTH);
287 0800 2
288 0801 2 ! if volume mounted then make the volume and the unit it is mounted on
289 0802 2 ! current. Else if the MTAACP is running in Automatic mode then all
290 0803 2 ! we need to do is get the next free drive. We must assume that if
291 0804 2 ! the drive has a vaild reel on it then it is the next reel the
292 0805 2 ! operator wishes us to use. If we are not running in Automatic mode this
293 0806 2 ! is not true and we must choose a unit, clear its previous use, and
294 0807 2 ! make the volume and the new unit current.
295 0808 2
296 0809 2 UCB_LIST = BBLOCK[.CURRENT_VCB[VCB$L_RVT], RVT$L_UCBLST];
297 0810 2 IF .MVL_ENTRY[MVL$V_MOUNTED] AND NOT .FLAGS[MOU$V_MOUNTERR]
298 0811 2 THEN KERNEL_CALL(MAKE_CUR_VOL, .MVL_ENTRY[MVL$B_RVN], .VOL)
299 0812 2 ELSE
300 0813 2 BEGIN
301 0814 2
302 0815 2 ! If we are running in Automatic mode then we want to unload the
303 0816 2 ! last volume so that the operator can put the next reel on the drive.
304 0817 2 ! However we also want to special case the fact that the user
305 0818 2 ! may have only one drive and thus force the operator to intervene.
306 0819 2
307 0820 2 IF NOT .CURRENT_VCB[VCB$V_NOAUTO]
308 0821 2 AND (.BBLOCK[.CURRENT_VCB[VCB$L_RVT], RVT$B_NVOLS] GTR 1)
309 0822 2 THEN
310 0823 2 BEGIN
311 0824 2 KERNEL_CALL(CLEAR_UNIT);
312 0825 2 KERNEL_CALL(MAKE_CUR_VOL, CHOOSE_UNIT(), .VOL);
313 0826 2 END
314 0827 2 ELSE KERNEL_CALL(CLPREV_MAKECUR, CHOOSE_UNIT(), .VOL);
315 0828 2
316 0829 2 END;
317 0830 2 ! now if the volume is mounted and no rewind is required just return
318 0831 2
319 0832 2 IF .MVL_ENTRY[MVL$V_MOUNTED]
320 0833 2 AND NOT .FLAGS[MOU$V_MOUNTERR]
321 0834 2 AND NOT .FLAGS[MOU$V_REWIND]
322 0835 2 THEN RETURN .MVL_ENTRY;
323 0836 2
```



```

324 0837 2 ! Assume that we won't send a message to the operators console. We would
325 0838 2 ! want to sent one if we switched to a new reel on a volume set without
326 0839 2 ! the operator getting involved in anyway. That is automatic volume
327 0840 2 ! recognition is turned on and no errors occurred while mounting the
328 0841 2 ! next volume. Also assume that no label will be specified.
329 0842 2
330 0843 2 INFORM OPER [0] = FALSE;
331 0844 2 LABEL_SPEC [0] = FALSE;
332 0845 2
333 0846 2 ! Call to get the device name and length of the name. These fields are
334 0847 2 ! stored in fields accessible to other routines in this module so that
335 0848 2 ! only one call need be done.
336 0849 2
337 0850 2 GET_DEV_NAME (CVT_DEVNAM_LENGTH, CVT_DEVNAM);
338 0851 2
339 0852 2 ! When the Mtaacp is running in Automtic mode, before asking the
340 0853 2 ! operator for the tape and label try to generate the label with
341 0854 2 ! in the ACP. And try to mount the volume on the given unit with
342 0855 2 ! that label.
343 0856 2
344 0857 2 IF NOT .CURRENT_VCB[VCB$V_NOAUTO]
345 0858 2 THEN
346 0859 2 BEGIN
347 0860 4 IF KERNEL_CALL(CREATE_LABEL, .VOL, .MVL)
348 0861 3 THEN
349 0862 4 BEGIN
350 0863 4
351 0864 4 ! Default the following fields
352 0865 4 ! LABEL_SPEC : We have a label for the next volume to mount.
353 0866 4 ! INFORM_OPER : If this mount works inform the operator that
354 0867 4 ! we have mounted a volume 'behind his back'.
355 0868 4
356 0869 4 LABEL_SPEC [0] = TRUE;
357 0870 4 INFORM_OPER [0] = TRUE;
358 0871 4
359 0872 4 ! If there is only one drive in this volume set then must give
360 0873 4 ! the operator some time to put the tape on the drive.
361 0874 4
362 0875 4 IF .BBLOCK [.CURRENT_VCB[VCB$L_RVT], RVT$B_NVOLS] GTR 1
363 0876 4 THEN
364 0877 5 BEGIN
365 0878 5 KERNEL_CALL(ASSUME_MOUNTED);
366 0879 5 KERNEL_CALL(SEND_ERRLOG,1,.CURRENT_UCB);
367 0880 4 END;
368 0881 3 END;
369 0882 2 END;
370 0883 2
371 0884 2 ! loop until we have a good mount
372 0885 2
373 0886 2 WHILE 1 DO
374 0887 2 BEGIN
375 0888 2
376 0889 2 LOCAL STATUS;
377 0890 2
378 0891 2 ! assume all is going to work
379 0892 2
380 0893 2 STATUS = TRUE;
```

381 0894 3
382 0895 3
383 0896 3
384 0897 3
385 0898 3
386 0899 4
387 0900 4
388 0901 4
389 0902 4
390 0903 4
391 0904 4
392 0905 4
393 0906 4
394 0907 4
395 0908 4
396 0909 4
397 0910 4
398 0911 4
399 0912 5
400 0913 5
401 0914 5
402 0915 5
403 0916 5
404 0917 5
405 0918 4
406 0919 4
407 0920 4
408 0921 4
409 0922 4
410 0923 4
411 0924 4
412 0925 4
413 0926 4
414 0927 4
415 0928 4
416 0929 4
417 0930 4
418 0931 5
419 0932 5
420 0933 5
421 0934 4
422 0935 4
423 0936 4
424 0937 4
425 0938 4
426 0939 4
427 0940 4
428 0941 4
429 0942 4
430 0943 4
431 0944 4
432 0945 4
433 0946 5
434 0947 5
435 0948 5
436 0949 5
437 0950 5

```
! Does the operator need to get involved ( mount a reel on a drive )
IF NOT .MVL_ENTRY[MVL$V_MOUNTED] OR .FLAGS[MOU$V_MOUNTERR]
THEN
  BEGIN
    LOCAL
      LABEL_ADDR : REF VECTOR [,BYTE],
      LABEL_SZ,
      MESSAGE_NUMBER;

    ! find the size to the label, do not print trailing spaces
    LABEL_ADDR = MVL_ENTRY [ MVL$T_VOLLBL ];
    IF .MVL_ENTRY [MVL$V_UNUSED]
    THEN LABEL_SZ = 0
    ELSE
      BEGIN
        LABEL_SZ = 6;
        DECR I FROM MVL$S_VOLLBL - 1 TO 0 DO
          IF .LABEL_ADDR [.I] NEQ ' '
          THEN EXIT[COOP]
          ELSE LABEL_SZ = .LABEL_SZ - 1;
        END;

      ! tell the operator to mount the reel
      IF .FLAGS[MOU$V_LBLCHECK]
      THEN MESSAGE_NUMBER = MOUN$ REMOUVOL
      ELSE IF .FLAGS[MOU$V_CHKIFSPC]
      THEN MESSAGE_NUMBER = MOUN$ MOUVOL
      ELSE MESSAGE_NUMBER = MOUN$ MOUNEWVOL;
      IF NOT PRINT OPR_MSG(.MESSAGE_NUMBER, .MAIL_CHANNEL,
        .CURRENT_VCB[VCB$B_CUR_RVN], .LABEL_SZ, .LABEL_ADDR,
        .CVT_DEVNAM_LENGTH, CVT_DEVNAM)
      THEN
        BEGIN
          KERNEL_CALL(TERMINATE VOL, .CURRENT_WCB);
          ERR_EXIT(SS$_NOTAPEOP);
        END;

      ! block the process and wait for the operator to reply
      ENABLE MAIL_AST();
      BLOCK(VCB$M_WAIMOUVOL);

      ! The operator has replied so check if a label has been specified
      ! If it has then check it to be sure that it is a valid ANSI label.
      IF .MAILSZ GTR 0
      THEN
        BEGIN
          ! stuff the label in MVL after doing ANSI checks
          STATUS = KERNEL_CALL(OPERATOR_LBL);
```



```

: 438      0951      5      IF NOT .STATUS
: 439      0952      5      THEN PRINT_OPR_MSG ( .STATUS, 0,
: 440      0953      5      .CVT_DEVNAM_LENGTH, CVT_DEVNAM)
: 441      0954      5      ELSE LABEL_SPEC [0] = TRUE;
: 442      0955      4      END;
: 443      0956      4
: 444      0957      4      ! No need to send another message to the operators console
: 445      0958      4
: 446      0959      4      INFORM_OPER [0] = FALSE;
: 447      0960      4
: 448      0961      4      ! assume device is mounted
: 449      0962      4
: 450      0963      4      KERNEL_CALL(ASSUME_MOUNTED);
: 451      0964      4      KERNEL_CALL(SEND_ERRLOG, 1, .CURRENT_UCB);
: 452      0965      3      END;
: 453      0966      3
: 454      0967      3      ! the reel was just mount or was already mounted now check it for
: 455      0968      3      being on online and valid
: 456      0969      3
: 457      0970      3      IF .STATUS
: 458      0971      3      THEN
: 459      0972      3
: 460      0973      3          ! Rewind the reel
: 461      0974      3
: 462      0975      3          INCRU J FROM 0 TO 29 DO
: 463      0976      4              BEGIN
: 464      0977      4                  STATUS = REWIND_AND_WAIT();
: 465      0978      4
: 466      0979      4                  ! if on_line, then exit loop
: 467      0980      4
: 468      0981      4                  IF .STATUS THEN EXITLOOP;
: 469      0982      4
: 470      0983      4                  ! wait one second if offline
: 471      0984      4
: 472      0985      4                  IF SYS$SETIMR(TIMEFN, SECONDS, 0, 0)
: 473      0986      4                  THEN SYS$WAITFR(TIMEFN);
: 474      0987      3                  END;
: 475      0988      3
: 476      0989      3      ! check for the write ring if needed
: 477      0990      3
: 478      0991      4      IF .STATUS AND ( NOT .BBLOCK[CURRENT_UCB[UCB$L_DEVCHAR],DEV$V_SWL] )
: 479      0992      4      THEN IF NOT (STATUS = CHECK_RING ())
: 480      0993      3          THEN PRINT_OPR_MSG ( MOUN$WRITLCK, 0,
: 481      0994      3          .CVT_DEVNAM_LENGTH, CVT_DEVNAM);
: 482      0995      3
: 483      0996      3      ! check the users privileges to write and read to the volume
: 484      0997      3
: 485      0998      3      IF .STATUS
: 486      0999      3      THEN
: 487      1000      4          BEGIN
: 488      1001      4
: 489      1002      4          ! assume device is mounted
: 490      1003      4
: 491      1004      4          KERNEL_CALL(ASSUME_MOUNTED);
: 492      1005      4
: 493      1006      4          ! exit if "/BLANK" on the reply command on a write next volume
: 494      1007      4          ! operation
```

```

495      1008 4
496      1009 6
497      1010 5
498      1011 5
499      1012 4
500      1013 5
501      1014 5
502      1015 5
503      1016 5
504      1017 5
505      1018 5
506      1019 5
507      1020 5
508      1021 5
509      1022 4
510      1023 4
511      1024 4
512      1025 4
513      1026 4
514      1027 4
515      1028 4
516      1029 4
517      1030 4
518      1031 4
519      1032 4
520      1033 4
521      1034 4
522      1035 4
523      1036 4
524      1037 4
525      1038 4
526      1039 4
527      1040 4
528      1041 4
529      1042 4
530      1043 4
531      1044 4
532      1045 4
533      1046 4
534      1047 4
535      1048 4
536      1049 4
537      1050 4
538      1051 4
539      1052 4
540      1053 4
541      1054 4
542      1055 4
543      1056 4
544      1057 4
545      1058 4
546      1059 4

!
IF ((.MAIL[OPCSW_MS_STATUS] EQL (OPCS_BLANKTAPE AND %X'FFFF'))
OR .CURRENT_VCB[VCBSV_BLANK] )
AND NOT ( .FLAGS [MOUSV_CHKIFSPC] OR .FLAGS [MOUSV_LBLCHECK] )
THEN
    BEGIN
        ! if the use writes the tape he has override privs
        !
        KERNEL_CALL( SET_MVL_OVERRIDE, TRUE);

        ! mount has succeeded exit "try till good mount" loop
        !
        EXITLOOP;
    END;

    ! now check for ANSI accessibility and VMS protection and
    ! exit the "try till good mount" loop in everything is OK
    !
    IF CHECK_ACCESS ( .FLAGS ) THEN EXITLOOP;

    END;

    ! mount did not work for some reason, force operator intervention
    !
    FLAGS = .FLAGS OR $FIELDMASK(MOUSV_MOUNTERR);

    ! reset the state of things
    !
    KERNEL_CALL(CLPREV_MAKECUR, .MVL_ENTRY[MVL$B_RVN], .VOL);
    END;
    ! end of while not good mount

! Check to see if the operator should hear about the switch then return.
IF .INFORM_OPER [0]
THEN
    BEGIN
        LOCAL
        DESCR : VECTOR [2];

        DESCR [0] = .CVT_DEVNAM_LENGTH;
        DESCR [1] = CVT_DEVNAM;

        ! Descr of the device name for
        ! the FAO field in the msg

        ! Length of dev name
        ! Address of the dev name

        ! Assume that the size of the label is 6. This is a safe assumption
        ! because we generated the label.

        PRINT_OPR_MSG (MOUN$_MOUNTED, 0, 6, MVL_ENTRY[MVL$T_VOLLBL],DESCR);
    END;
    RETURN .MVL_ENTRY;

END;

! end of routine MOUNT_VOL
```

```

.TITLE MOUVOL
.IDENT \V04-000\
```



```
00 00 41 31 31 45 4C 49 46 43 45 44 00000 P.AAA: .ASCII \DECFILE11A\<0><0>
      FFFFFFFF FF676980 0000C P.AAB: .LONG -10000000, -1
```

```
.PSECT $LOCKEDD1$,NOEXE,2
```

```
00000 TAPE_OWNER:
      .BLKB 4
00004 TAPE_PROT:
      .BLKB 2
00006 .BLKB 2
00008 LABEL_SPEC:
      .BLKB 1
00009 .BLKB 3
0000C INFORM_OPER:
      .BLKB 1
0000D .BLKB 3
00010 CVT_DEVNAM:
      .BLKB 16
00020 CVT_DEVNAM_LENGTH:
      .BLKB 1
```

```
STARID=
SECONDS=
```

```
P.AAA
```

```
P.AAB
```

```
.EXTRN CURRENT_UCB, CURRENT_WCB
.EXTRN IO_CHANNEL, IO_STATUS
.EXTRN MAIL_CHANNEL, WORK_AREA
.EXTRN BLOCK, CHECK_PROT
.EXTRN ENABLE_MAIL_AST
.EXTRN GET_CCB, GET_RECORD
.EXTRN IOC$CVT_DEVNAM, PRINT_OPR_MSG
.EXTRN PRINT_NOT_LABEL
.EXTRN PROCESS_VOL2_LABEL
.EXTRN READ_BLOCK, REWIND_AND_WAIT
.EXTRN SEND_ERRLOG, SYSS$FIO
.EXTRN SYSS$QIOW, SYSS$SETIMR
.EXTRN SYSS$WAITFR, TAPE_OWN_PROT
.EXTRN TERMINATE_VOL, SYSS$CMKRNL
```

```
.PSECT $CODE$,NOWRT,2
```

```
52      0B      A4      07FC 00000
58      0000V CF 9E 00002
57      0000G CF 9E 00007
56      0000' CF 9E 0000C
55 00000000G 9F 9E 00011
5E      08 C2 00018
54      34 AB D0 0001B
52      04 AC D0 0001F
08      00 ED 00023
      10 18 00029
      14 BB 0002B
      02 DD 0002D
      5E DD 0002F
      CF 9F 00031
65      0000V 05 FB 00035

.ENTRY MOUNT_VOL, Save R2,R3,R4,R5,R6,R7,R8,R9,R10 ; 0723
MOVAB ASSUME_MOUNTED, R8
MOVAB PRINT_OPR_MSG, R7
MOVAB CVT_DEVNAM, R6
MOVAB @SYSS$CMKRNL, R5
SUBL2 #8, SP
MOVL 52(CURRENT_VCB), MVL ; 0792
MOVL VOL, R2 ; 0793
CMPZV #0, #8, 11(MVL), R2
BGEQ 1$
PUSHR #*M<R2,R4> ; 0795
PUSHL #2
PUSHL SP
PUSHAB MAKE_VOL_ENTRY
CALLS #5, SYSS$CMKRNL
```

54		50	D0	00038	MOVL	R0, MVL	
59	1C	A442	7E	0003B	1\$:	MOVAB	28(MVL)[R2], MVL_ENTRY
53	20	AB	D0	00040		MOVAB	32(CURRENT_VCB), -R3
5A	44	A3	9E	00044		BLBC	68(R3), UCB_LIST
0D	07	A9	E9	00048		BBS	7(MVL_ENTRY), 2\$
08	08	AC	03	E0	0004C	BBS	#3, FLAGS, 2\$
			52	DD	00051	PUSHL	R2
		7E	06	A9	9A	00053	MOVZBL
			1D	11	00057	BRB	6(MVL_ENTRY), -(SP)
22	2D	AB	04	E0	00059	2\$:	BBS
		01	0B	A3	91	0005E	CMPB
			1C	1B	00062	BLEQU	11(R3), #1
			7E	D4	00064	CLRL	4\$
			5E	DD	00066	PUSHL	-(SP)
		0000V	CF	9F	00068	PUSHL	SP
		65	03	FB	0006C	PUSHAB	CLEAR_UNIT
			52	DD	0006F	CALLS	#3, SYS\$CMKRNL
		0000V	30	00071	PUSHL	R2	
			50	DD	00074	BSBW	CHOOSE_UNIT
			02	DD	00076	3\$:	PUSHL
			5E	DD	00078	PUSHL	R0
		0000V	CF	9F	0007A	PUSHL	#2
			0F	11	0007E	PUSHL	SP
			52	DD	00080	4\$:	PUSHAB
		0000V	30	00082	PUSHL	MAKE_CUR_VOL	
			50	DD	00085	BRB	5\$
			02	DD	00087	PUSHL	R2
			5E	DD	00089	BSBW	CHOOSE_UNIT
		0000V	CF	9F	0008B	PUSHL	R0
		65	05	FB	0008F	5\$:	PUSHL
		0C	07	A9	E9	00092	PUSHL
		AC	03	E0	00096	PUSHAB	CLPREV_MAKECUR
		03	08	AC	E8	0009B	CALLS
			01F7	31	0009F	6\$:	#5, SYS\$CMKRNL
		FC	01	8A	000A2	BLBC	7(MVL_ENTRY), 6\$
		F8	01	8A	000A6	BBS	#3, FLAGS, 6\$
			56	DD	000AA	BLBS	FLAGS, 6\$
			A6	9F	000AC	BRW	29\$
		0000V	02	FB	000AF	BICB2	#1, INFORM_OPER
		2D	04	E0	000B4	BICB2	#1, LABEL_SPEC
			14	BB	000B9	PUSHL	R6
			02	DD	000BB	PUSHAB	CVT_DEVNAM_LENGTH
			5E	DD	000BD	CALLS	#2, GET_DEV_NAME
		65	CF	9F	000BF	BBS	#4, 45(CURRENT_VCB), 8\$
		2C	05	FB	000C3	PUSHR	#^M<R2,R4>
		A6	50	E9	000C6	PUSHL	#2
		A6	01	88	000C9	PUSHL	SP
		50	01	88	000CD	PUSHAB	CREATE_LABEL
		01	20	AB	D0	000D1	CALLS
			0B	A0	91	000D5	#5, SYS\$CMKRNL
				1A	1B	000D9	BLBC
				7E	D4	000DB	R0, 8\$
		4100	8F	BB	000DD	BISB2	#1, LABEL_SPEC
		65	03	FB	000E1	BISB2	#1, INFORM_OPER
		0000G	CF	DD	000E4	MOVL	32(CURRENT_VCB), R0
			01	DD	000E8	CMPB	11(R0), #1
			02	DD	000EA	BLEQU	8\$
						CLRL	-(SP)
						PUSHR	#^M<R8,SP>
						CALLS	#3, SYS\$CMKRNL
						PUSHL	CURRENT_UCB
						PUSHL	#1
						PUSHL	#2

			5E	DD	000EC	PUSHL	SP		
			CF	9F	000EE	PUSHAB	SEND_ERRLOG		
	65	0000G	05	FB	000F2	CALLS	#5, SYSSCMKRNL		
	53		01	DO	000F5	7\$:			
	08		A9	E9	000F8	8\$:			0893
03	08		03	E0	000FC	BLBC	7(MVL_ENTRY), 9\$		0897
			00C5	31	00101	BBS	#3, FLAGS, 9\$		
			59	DO	00104	BRW	19\$		
04	07		01	E1	00107	9\$:	MOV_L	MOV_L_ENTRY, LABEL_ADDR	0908
			52	D4	0010C	BBC	#1, 7(MVL_ENTRY), 10\$		0909
			11	11	0010E	CLRL	LABEL_SZ		0910
			06	DO	00110	BRB	12\$		
	52		05	DO	00113	10\$:	MOV_L	#6, LABEL_SZ	0913
	51		91	DO	00116	11\$:	MOV_L	#5, I	0915
	20		05	12	0011A	CMPB	(I)[LABEL_ADDR], #32		
			52	D7	0011C	BNEQ	12\$		
			51	F4	0011E	DECL	LABEL_SZ		0917
09	08		01	E1	00121	SOBGEQ	I, 11\$		0915
			8F	DO	00126	BBC	#1, FLAGS, 13\$		0922
		0072820C	15	11	0012D	MOV_L	#7504396, MESSAGE_NUMBER		0923
09	08		02	E1	0012F	BRB	15\$		
			8F	DO	00134	BBC	#2, FLAGS, 14\$		0924
			07	11	0013B	MOV_L	#7504028, MESSAGE_NUMBER		0925
			8F	DO	0013D	BRB	15\$		
		00728204	56	DD	00144	14\$:	MOV_L	#7504388, MESSAGE_NUMBER	0926
			50	DD	0014A	15\$:	PUSHL	R6	0927
	7E	10	52	DD	0014C	MOVZBL	CVT_DEVNAM_LENGTH, -(SP)		0929
			AB	9A	0014E	PUSHL	LABEL_ADDR		0928
			CF	DD	00152	PUSHL	LABEL_SZ		
			51	DD	00156	MOVZBL	47(CURRENT_VCB), -(SP)		
			67	16	00158	PUSHL	MAIL_CHANNEL		0927
			1C	C0	0015A	PUSHL	MESSAGE_NUMBER		
	5E		50	E8	0015D	JSB	PRINT_OPR_MSG		
	13		CF	DD	00160	ADDL2	#28, SP		
			01	DD	00164	BLBS	R0, 16\$		
			5E	DD	00166	PUSHL	CURRENT_WCB		0932
			CF	9F	00168	PUSHL	#1		
	65		04	FB	0016C	PUSHL	SP		
			8F	BF	0016F	PUSHAB	TERMINATE_VOL		
0000G	CF	0264	00	FB	00173	CALLS	#4, SYSSCMKRNL		0933
			04	DD	00178	CHMU	#612		0938
0000G	CF		01	FB	0017A	CALLS	#0, ENABLE_MAIL_AST		0939
			CF	D5	0017F	PUSHL	#4		
			26	15	00183	CALLS	#1, BLOCK		
			7E	D4	00185	TSTL	MAILSZ		0944
			5E	DD	00187	BLEQ	18\$		
			CF	9F	00189	CLRL	-(SP)		0950
			03	FB	0018D	PUSHL	SP		
	65		50	DO	00190	PUSHAB	OPERATOR_LBL		
	53		53	E8	00193	CALLS	#3, SYSSCMKRNL		
	11		56	DD	00196	MOV_L	R0, STATUS		0951
			7E	D4	0019C	BLBS	STATUS, 17\$		0952
			53	DD	0019E	PUSHL	R6		0953
			67	16	001A0	MOVZBL	CVT_DEVNAM_LENGTH, -(SP)		0952
	5E		10	C0	001A2	CLRL	-(SP)		
						PUSHL	STATUS		
						JSB	PRINT_OPR_MSG		
						ADDL2	#16, SP		

			04	11	001A5	BRB	18\$		
	FB	A6	01	88	001A7	BISB2	#1, LABEL_SPEC		0954
	FC	A6	01	8A	001AB	BICB2	#1, INFORM_OPER		0959
			7E	D4	001AF	CLRL	-(SP)		0963
		65	8F	BB	001B1	PUSHR	#^M<R8,SP>		
			03	FB	001B5	CALLS	#3, SYSSCMKRNL		
		0000G	CF	DD	001B8	PUSHL	CURRENT_UCB		0964
			01	DD	001BC	PUSHL	#1		
			02	DD	001BE	PUSHL	#2		
			5E	DD	001C0	PUSHL	SP		
		0000G	CF	9F	001C2	PUSHAB	SEND_ERRLOG		
			05	FB	001C6	CALLS	#5, SYSSCMKRNL		
		65	53	E9	001C9	BLBC	STATUS, 23\$		0970
		5A	52	D4	001CC	CLRL	J		0975
	0000G		00	FB	001CE	CALLS	#0, REWIND_AND_WAIT		0977
			50	DD	001D3	MOVL	R0, STATUS		
			53	E8	001D6	BLBS	STATUS, 22\$		0981
			7E	7C	001D9	CLRL	-(SP)		0985
		FE19	CF	9F	001DB	PUSHAB	SECONDS		
			03	DD	001DF	PUSHL	#3		
	00000000G		04	FB	001E1	CALLS	#4, @NSYSSSETIMR		
			50	E9	001E8	BLBC	R0, 21\$		
			03	DD	001EB	PUSHL	#3		0986
	00000000G		01	FB	001ED	CALLS	#1, @NSYSSWAITFR		
			52	D6	001F4	INCL	J		0975
			52	D1	001F6	CMPL	J, #29		
			D3	1B	001F9	BLEQU	20\$		
			53	E9	001FB	BLBC	STATUS, 27\$		0991
		0000G	CF	DD	001FE	MOVL	CURRENT_UCB, R0		
1E			01	EO	00203	BBS	#1, 59(R0), 23\$		
	3B		00	FB	00208	CALLS	#0, CHECK_RING		0992
	0000V		50	DD	0020D	MOVL	R0, STATUS		
			53	E8	00210	BLBS	STATUS, 24\$		
			56	DD	00213	PUSHL	R6		0993
			A6	9A	00215	MOVZBL	CVT DEVNAM_LENGTH, -(SP)		0994
			7E	D4	00219	CLRL	-(SP)		0993
		00728134	8F	DD	0021B	PUSHL	#7504180		
			67	16	00221	JSB	PRINT_OPR_MSG		
			10	CO	00223	ADDL2	#16, SP		
			53	E9	00226	BLBC	STATUS, 27\$		0998
			7E	D4	00229	CLRL	-(SP)		1004
		4100	8F	BB	0022B	PUSHR	#^M<R8,SP>		
			03	FB	0022F	CALLS	#3, SYSSCMKRNL		
	81E3		CF	B1	00232	CMPL	MAIL+2, #33251		1009
			05	13	00239	BEQL	25\$		
			02	E1	0023B	BBC	#2, 45(CURRENT_VCB), 26\$		1010
			02	EO	00240	BBS	#2, FLAGS, 26\$		1011
			01	EO	00245	BBS	#1, FLAGS, 26\$		
			01	DD	0024A	PUSHL	#1		1017
			01	DD	0024C	PUSHL	#1		
			5E	DD	0024E	PUSHL	SP		
		0000V	CF	9F	00250	PUSHAB	SET_MVL_OVERRIDE		
			04	FB	00254	CALLS	#4, SYSSCMKRNL		
			21	11	00257	BRB	28\$		1013
		08	AC	DD	00259	PUSHL	FLAGS		1027
	0000V		01	FB	0025C	CALLS	#1, CHECK_ACCESS		
			50	E8	00261	BLBS	R0, 28\$		

MOUVOL
V04-000

L 10
16-Sep-1984 02:25:33
14-Sep-1984 12:46:44

VAX-11 Bliss-32 V4.0-742
[MTAACP.SRC]MOUVOL.B32;1

Page 15
(2)

08	AC		08	88	00264	27\$:	BISB2	#8, FLAGS	:	1033
		04	AC	DD	00268		PUSHL	VOL	:	1037
	7E	06	A9	9A	0026B		MOVZBL	6(MVL_ENTRY), -(SP)	:	
			02	DD	0026F		PUSHL	#2	:	
			5E	DD	00271		PUSHL	SP	:	
		0000V	CF	9F	00273		PUSHAB	CLPREV_MAKECUR	:	
			FE78	31	00277		BRW	7\$:	
	1B	FC	A6	E9	0027A	28\$:	BLBC	INFORM OPER, 29\$:	1042
	6E	10	A6	9A	0027E		MOVZBL	CVT_DEVNAM_LENGTH, DESCR	:	1049
04	AE		66	9E	00282		MOVAB	CVT_DEVNAM, DESCR+4	:	1050
		4200	8F	BB	00286		PUSHR	#^MZR9, SP>	:	1055
			06	DD	0028A		PUSHL	#6	:	
			7E	D4	0028C		CLRL	-(SP)	:	
		0072A003	8F	DD	0028E		PUSHL	#7512067	:	
	5E		67	16	00294		JSB	PRINT_OPR_MSG	:	
	50		14	C0	00296		ADDL2	#20, SP	:	
			59	D0	00299	29\$:	MOVL	MVL_ENTRY, R0	:	1057
			04	00	0029C		RET		:	1059

; Routine Size: 669 bytes, Routine Base: \$CODE\$ + 0014

```
548 1060 1 GLOBAL ROUTINE GET_DEV_NAME(LEN, C_DEVN) : COMMON_CALL NOVALUE =
549 1061 1
550 1062 1 ++
551 1063 1
552 1064 1 FUNCTIONAL DESCRIPTION:
553 1065 1 This routine converts a UCB address into te device's name by calling
554 1066 1 the system routine IOC$CVT_DEVANM.
555 1067 1
556 1068 1 CALLING SEQUENCE:
557 1069 1 GET_DEV_NAME(ARG1,ARG2)
558 1070 1
559 1071 1 INPUT PARAMETERS:
560 1072 1 ARG1 - Address to store the length of the device name
561 1073 1 ARG2 - Address to store the name of the device
562 1074 1
563 1075 1 IMPLICIT INPUTS:
564 1076 1 CURRENT_UCB - address of current unit control block
565 1077 1
566 1078 1 OUTPUT PARAMETERS:
567 1079 1 The length of the device name and the device name are returned
568 1080 1
569 1081 1 IMPLICIT OUTPUTS:
570 1082 1 NONE
571 1083 1
572 1084 1 ROUTINE VALUE:
573 1085 1 NONE
574 1086 1
575 1087 1 SIDE EFFECTS:
576 1088 1 NONE
577 1089 1
578 1090 1 --
579 1091 1
580 1092 2 BEGIN
581 1093 2
582 1094 2 EXTERNAL REGISTER
583 1095 2 COMMON_REG;
584 1096 2
585 1097 2 BIND
586 1098 2 CVT_DEVNAME = .C_DEVN, ! Storage for cvted device name
587 1099 2 LENGTH = .LEN, ! Length of device name
588 1100 2 IN_NAME_LENGTH = MAX_DEVNAM_LENGTH; ! Set the maxium lentgh if dev nam
589 1101 2
590 1102 2 LOCAL
591 1103 2 OUT_NAME_LENGTH : BYTE, ! Actual length of device name gotten from the convert
592 1104 2 CVT_DEVNAM_STATUS, ! Status of the convert on devnam
593 1105 2 DEV_NAME : VECTOR [MAX_DEVNAM_LENGTH,BYTE]; ! Local storage for the
594 1106 2 ! name from the convert
595 1107 2
596 1108 2 ! Call to the system routine to get the device name of the drive to use
597 1109 2 ! given the UCB associated with that drive. This routine gets both
598 1110 2 ! the node name and the device name.
599 1111 2 ! NOTE: DEV_NAME must be in local storage, because the routine expects
600 1112 2 ! this field to be always accessible.
601 1113 2
602 1114 2 CVT_DEVNAM_STATUS = IOC$CVT_DEVNAM ( IN_NAME_LENGTH, DEV_NAME, 0,
603 1115 2 .CURRENT_UCB; OUT_NAME_LENGTH);
604 1116 2
```


MOUVOL
V04-000

N 10

16-Sep-1984 02:25:33

14-Sep-1984 12:46:44

VAX-11 Bliss-32 V4.0-742

[MTAACP.SRC]MOUVOL.B32;1

Page 17

(3)

```
: 605      1117  2  ! Move the resultant string into a field accessible to the entire module.
: 606      1118  2  ! Also fill the first byte of the string with the size of the string for
: 607      1119  2  ! the call to PRINT_OPR_MSG. The FAO string expects the device name to be
: 608      1120  2  ! in this format.
: 609      1121  2
: 610      1122  2
: 611      1123  2
: 612      1124  1  CH$MOVE (.OUT_NAME_LENGTH,DEV_NAME,CVT_DEVNAME);
                     LENGTH = .OUT_NAME_LENGTH;
                     END;
```

IN_NAME_LENGTH= 16

			007C 00000	.ENTRY	GET_DEV_NAME, Save R2,R3,R4,R5,R6	: 1060
5E		10	C2 00002	SUBL2	#16, SP	: 1114
51		6E	9E 00005	MOVAB	DEV_NAME, R1	
55	0000G	CF	D0 00008	MOVL	CURRENT_UCB, R5	
		54	D4 0000D	CLRL	R4	
50		10	D0 0000F	MOVL	#16, R0	
	00000000G	9F	16 00012	JSB	@#I0C\$CVT DEVNAM	
56		51	D0 00018	MOVL	R1, OUT_NAME_LENGTH	: 1122
50		56	9A 0001B	MOVZBL	OUT_NAME_LENGTH, R0	
08 BC		50	28 0001E	MOV3	R0, DEV_NAME, @C_DEVN	: 1123
	04 BC	56	9A 00023	MOVZBL	OUT_NAME_LENGTH, @LEN	: 1124
		04	00027	RET		

: Routine Size: 40 bytes, Routine Base: \$CODE\$ + 02B1

```
: 613      1125  1
: 614      1126  1
```

```

: 616      1127 1 ROUTINE CHOOSE_UNIT : L$CHOOSE_UNIT =
: 617      1128 1
: 618      1129 1 ++
: 619      1130 1
: 620      1131 1 FUNCTIONAL DESCRIPTION:
: 621      1132 1     this routine chooses the next unit to use
: 622      1133 1
: 623      1134 1 CALLING SEQUENCE:
: 624      1135 1     CHOOSE_UNIT()
: 625      1136 1
: 626      1137 1 INPUT PARAMETERS:
: 627      1138 1     NONE
: 628      1139 1
: 629      1140 1 IMPLICIT INPUTS:
: 630      1141 1     CURRENT_VCB - address of current volume control block
: 631      1142 1
: 632      1143 1 OUTPUT PARAMETERS:
: 633      1144 1     NONE
: 634      1145 1
: 635      1146 1 IMPLICIT OUTPUTS:
: 636      1147 1     NONE
: 637      1148 1
: 638      1149 1 ROUTINE VALUE:
: 639      1150 1     relative unit number
: 640      1151 1
: 641      1152 1 SIDE EFFECTS:
: 642      1153 1     The current algorithm is increment current unit.
: 643      1154 1     If it is allocated then use it else wrap around to first unit.
: 644      1155 1
: 645      1156 1 --
: 646      1157 1
: 647      1158 2 BEGIN
: 648      1159 2
: 649      1160 2 EXTERNAL REGISTER
: 650      1161 2     COMMON_REG;
: 651      1162 2
: 652      1163 2 LOCAL
: 653      1164 2     NUNITS,           ! number of units allocated
: 654      1165 2     RVT      : REF BBLOCK, ! address of unit table
: 655      1166 2     UNIT;          ! unit to use
: 656      1167 2
: 657      1168 2     RVT = .CURRENT_VCB[VCB$L_RVT]; ! get address of unit table
: 658      1169 2
: 659      1170 2     NUNITS = .RVT[RVT$B_NVOLS]; ! get # of units allocated to
: 660      1171 2     ! this volume set
: 661      1172 2     UNIT = .CURRENT_VCB[VCB$W_RVN] + 1; ! inc current rel unit number
: 662      1173 2
: 663      1174 2     IF .UNIT GEQ .NUNITS
: 664      1175 2     THEN
: 665      1176 2         UNIT = 0; ! if not allocated, use 1st one
: 666      1177 2
: 667      1178 2     RETURN .UNIT;
: 668      1179 2
: 669      1180 1     END; ! end of routine CHOOSE_UNIT
```


C 11
16-Sep-1984 02:25:33 VAX-11 BLISS-32 V4.0-742
14-Sep-1984 12:46:44 [MTAACP.SRC]MOUVOL.B32;1

Page 19
(4)

Address	Op Code	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
---------	---------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

```
; Routine Size: 22 bytes,    Routine Base: $CODE$ + 02D9
```

MOU
V04

```

: 671 1181 1 ROUTINE CLEAR_UNIT : NOVALUE MVL_UCB =
: 672 1182 1
: 673 1183 1 ++
: 674 1184 1
: 675 1185 1 FUNCTIONAL DESCRIPTION:
: 676 1186 1 This routine dismounts the last used volume in a volume set. It does
: 677 1187 1 this by checking the Current UCB against the list of UCB's to find the
: 678 1188 1 unit number. Then checking each entry in the MVL for that unit number.
: 679 1189 1 If that unit is marked mounted then a QIO is issued to dismount and
: 680 1190 1 unload that volume and the MVL is marked dismounted. This will allow
: 681 1191 1 the operator to mount the next reel in the volume set before it is
: 682 1192 1 actually needed by MTAACP processing.
: 683 1193 1
: 684 1194 1 CALLING SEQUENCE:
: 685 1195 1 CLEAR_UNIT() called in Kernel mode
: 686 1196 1
: 687 1197 1 INPUT PARAMETERS:
: 688 1198 1 NONE
: 689 1199 1
: 690 1200 1 IMPLICIT INPUTS:
: 691 1201 1 CURRENT_VCB - address of current volume control block
: 692 1202 1
: 693 1203 1 OUTPUT PARAMETERS:
: 694 1204 1 NONE
: 695 1205 1
: 696 1206 1 IMPLICIT OUTPUTS:
: 697 1207 1 Last mounted volume is dismount and unloaded.
: 698 1208 1
: 699 1209 1 ROUTINE VALUE:
: 700 1210 1 none
: 701 1211 1
: 702 1212 1 SIDE EFFECTS:
: 703 1213 1 none
: 704 1214 1
: 705 1215 1
: 706 1216 1 --
: 707 1217 1
: 708 1218 2 BEGIN
: 709 1219 2
: 710 1220 2 EXTERNAL REGISTER
: 711 1221 2 MVL_ENTRY = 9 : REF BBLOCK, ! Address of current MVL not
: 712 1222 2 ! used by this routine
: 713 1223 2 UCB_LIST = 10 : REF VECTOR, ! Addr of UCB list for vol set
: 714 1224 2 COMMON_REG;
: 715 1225 2
: 716 1226 2 LOCAL
: 717 1227 2 NUNITS, ! number of units allocated
: 718 1228 2 RVT : REF BBLOCK, ! address of unit table
: 719 1229 2 UNIT, ! unit currently in use
: 720 1230 2 NVOLS, ! number of volumes in set
: 721 1231 2 MVL : REF BBLOCK, ! addr of magtape volume list
: 722 1232 2 MVL_ADDR : REF BBLOCKVECTOR [10,MVL$K_LENGTH]; ! addr of MVL control block
: 723 1233 2
: 724 1234 2
: 725 1235 2
: 726 1236 2 RVT = .CURRENT_VCB[VCB$L_RVT]; ! get address of unit table
: 727 1237 2 NUNITS = .RVT[RVT$B_NVOLS]; ! get # of units allocated to
```



```
728 1238 2 UNIT = 0; ! Set first unit
729 1239 2
730 1240 2 ! Check thru the list of UCB to find the unit number of the UCB currently
731 1241 2 ! in use.
732 1242 2
733 1243 2 INCR I FROM 0 TO .NUNITS - 1 DO
734 1244 2 IF .UCB_LIST [ .UNIT ] EQL .CURRENT_UCB
735 1245 2 THEN EXITLOOP
736 1246 2 ELSE UNIT = .UNIT + 1;
737 1247 2 MVL = .CURRENT_VCB[VCBSL_MVL]; ! get the magtape vol list addr
738 1248 2 NVOLS = .MVL[MVLSB_NVOLS]; ! get the numb of volume in set
739 1249 2 MVL_ADDR = .MVL + MVLSK_FIXLEN; ! get the first volume entry
740 1250 2
741 1251 2 ! Check thru the mounted volume labels to find the entry which points
742 1252 2 ! at the current unit and is mark as mounted. When we find that entry,
743 1253 2 ! a message is send to the ERRORLOG saying the volume has been dismounted;
744 1254 2 ! the entry is marked dismounted and a QIO is issued to the drive to unload
745 1255 2 ! the reel from the drive.
746 1256 2
747 1257 2 INCR I FROM 0 TO .NVOLS - 1 DO
748 1258 2 IF .MVL_ADDR [ .I, MVLSV_MOUNTED ]
749 1259 2 AND
750 1260 2 .UNIT EQL .MVL_ADDR [ .I, MVLSB_RVN ]
751 1261 2 THEN
752 1262 2 BEGIN
753 1263 2 KERNEL CALL(SEND_ERRLOG,0,.UCB_LIST[.UNIT]);
754 1264 2 MVL_ADDR [ .I, MVLSV_MOUNTED ] = 0;
755 1265 2 SYSSQIOW ( 0, .IO_CHANNEL,
756 1266 2 IOS_UNLOAD OR IOSM_NOWAIT OR IOSM_CLSEREXCP,
757 1267 2 0, 0, 0, 0, 0, 0, 0, 0);
758 1268 2 EXITLOOP;
759 1269 2 END;
760 1270 1 END;
```

003C 00000 CLEAR_UNIT:									
					.WORD	Save R2,R3,R4,R5			1181
	50	20	AB	D0	00002	MOVL	32(CURRENT_VCB), RVT		1236
	51	0B	A0	9A	00006	MOVZBL	11(RVT), NONITS		1237
			54	D4	0000A	CLRL	UNIT		1238
	50		01	CE	0000C	MNEGL	#1, I		1244
			0A	11	0000F	BRB	2\$		
	0000G	CF	6A44	D1	00011	1\$:	CMPL	(UCB_LIST)[UNIT], CURRENT_UCB	
			06	13	00017	BEQL	3\$		
			54	D6	00019	INCL	UNIT		1246
F2	50		51	F2	0001B	2\$:	AOBLSS	NUNITS, I, 1\$	1244
	50	34	AB	D0	0001F	3\$:	MOVL	52(CURRENT_VCB), MVL	1247
	55	0B	A0	9A	00023	MOVZBL	11(MVL), NVOLS		1248
	52	24	A0	9E	00027	MOVAB	36(R0), MVL_ADDR		1249
	53		01	CE	0002B	MNEGL	#1, I		1258
			4A	11	0002E	BRB	5\$		
		07	A243	7F	00030	4\$:	PUSHAQ	7(MVL_ADDR)[I]	
42	9E		00	E1	00034	BBC	#0, @SP)+, 5\$		
		06	A243	7F	00038	PUSHAQ	6(MVL_ADDR)[I]		1260

54	9E	08	00	ED	0003C	CMPZV	#0, #8, @ (SP)+, UNIT	:		
			37	12	00041	BNEQ	5\$:		
			6A44	DD	00043	PUSHL	(UCB_LIST)[UNIT]	:	1263	
		7E	02	7D	00046	MOVQ	#2, -(SP)	:		
			5E	DD	00049	PUSHL	SP	:		
	00000000G	9F	0000G	CF	9F	0004B	PUSHAB	SEND_ERRLOG	:	
			05	FB	0004F	CALLS	#5, @#SYSS\$CMKRN	:		
		9E	07	A243	7F	00056	PUSHAQ	7(MVL_ADDR)[I]	:	1264
			01	8A	0005A	BICB2	#1, @ (SP)+	:		
			7E	7C	0005D	CLRQ	-(SP)	:	1265	
			7E	7C	0005F	CLRQ	-(SP)	:		
			7E	7C	00061	CLRQ	-(SP)	:		
			7E	7C	00063	CLRQ	-(SP)	:		
			7E	D4	00065	CLRL	-(SP)	:		
		7E	0281	8F	3C	00067	MOVZWL	#641, -(SP)	:	1266
			0000G	CF	DD	0006C	PUSHL	IO_CHANNEL	:	1265
			7E	D4	00070	CLRL	-(SP)	:		
	00000000G	9F	0C	FB	00072	CALLS	#12, @#SYSS\$QIOW	:		
				04	00079	RET		:	1262	
	B2	53	55	F2	0007A	AOBLSS	NVOLS, I, 4\$:	1258	
			04	0007E	RET			:	1270	

; Routine Size: 127 bytes, Routine Base: \$CODE\$ + 02EF

; 761 1271 1


```

763 1272 1 ROUTINE CLPREV_MAKECUR (UNIT, VOL) : MVL_UCB NOVALUE =
764 1273 1
765 1274 1 ++
766 1275 1
767 1276 1 FUNCTIONAL DESCRIPTION:
768 1277 1 This routine clears the previous use of this unit in the
769 1278 1 mag tape volume list and makes the unit and rel volume current
770 1279 1
771 1280 1 CALLING SEQUENCE:
772 1281 1 CLPREV_MAKECUR(ARG1,ARG2), CALLED IN KERNEL MODE
773 1282 1
774 1283 1 INPUT PARAMETERS:
775 1284 1 ARG1 - relative unit number
776 1285 1 ARG2 - relative volume number
777 1286 1
778 1287 1 IMPLICIT INPUTS:
779 1288 1 CURRENT_VCB - address of current volume control block
780 1289 1 MVL_ENTRY - address of current entry in mvl for volume
781 1290 1 UCB_LIST - address of ucb's available for this unit
782 1291 1
783 1292 1 OUTPUT PARAMETERS:
784 1293 1 NONE
785 1294 1
786 1295 1 IMPLICIT OUTPUTS:
787 1296 1 NONE
788 1297 1
789 1298 1 ROUTINE VALUE:
790 1299 1 NONE
791 1300 1
792 1301 1 SIDE EFFECTS:
793 1302 1 if volume currently mounted on unit, rewind issued
794 1303 1
795 1304 1 --
796 1305 1
797 1306 2 BEGIN
798 1307 2
799 1308 2 EXTERNAL REGISTER
800 1309 2 MVL_ENTRY = 9 : REF BBLOCK, ! address of current rel vol in mvl
801 1310 2 UCB_LIST = 10 : REF VECTOR, ! address of UCB's
802 1311 2 COMMON_REG;
803 1312 2
804 1313 2 LOCAL
805 1314 2 CCB : REF BBLOCK, ! addr of ACP IO channel
806 1315 2 : control block
807 1316 2 MVL : REF BBLOCK, ! addr magtape volume list
808 1317 2 MVL_ADDR : REF BBLOCKVECTOR [10, MVL$K_LENGTH], ! address of MVL entries
809 1318 2
810 1319 2 NVOLS;
811 1320 2
812 1321 2
813 1322 2 CCB = GET CCB ( .IO CHANNEL ); ! calc ACP IO CCB
814 1323 2 MVL = .CURRENT_VCB[VCBSL_MVL]; ! address of mvl control block
815 1324 2 NVOLS = .MVL[MVL$B_NVOLS]; ! number of relative volume
816 1325 2 MVL_ADDR = .MVL + MVL$K_FIXLEN; ! addr of entries for rel vol
817 1326 2
818 1327 2 INCR I FROM 0 TO .NVOLS - 1 DO ! check each relative volume
819 1328 2 BEGIN
```

```

820      1329      3
821      1330      3
822      1331      3
823      1332      3
824      1333      3
825      1334      3
826      1335      4
827      1336      4
828      1337      4
829      1338      4
830      1339      4
831      1340      4
832      1341      4
833      1342      4
834      1343      4
835      1344      4
836      1345      4
837      1346      4
838      1347      4
839      1348      4
840      1349      3
841      1350      3
842      1351      2
843      1352      2
844      1353      2
845      1354      1

IF .MVL_ADDR[.I, MVL$V_MOUNTED]
AND
    .UNIT EQL .MVL_ADDR[.I, MVL$B_RVN] ! if mounted
! on unit about to be used
THEN
BEGIN
    KERNEL_CALL(SEND_ERRLOG, 0, .UCB_LIST[.UNIT]);
    MVL_ADDR[.I, MVL$V_MOUNTED] = 0; ! before mounted is cleared!!
! mark it dismounted,
! assign channel to it's unit,
!
CCB[CCB$U_UCB] = .UCB_LIST[.UNIT];
SYSSQIOW(0, .IO_CHANNEL,
    IOS_UNLOAD OR IOSM_NOWAIT OR IOSM_CLSEREXCP,
    0,0,0,0,0,0,0,0,0,0); ! unload the tape, please
EXITLOOP;
END;

END;

MAKE_CUR_VOL(.UNIT, .VOL);
END;

! end of routine CLPREV_MAKECUR
```

```

                                007C 00000 CLPREV_MAKECUR:
                                .WORD Save R2,R3,R4,R5,R6
                                0000G CF DD 00002 PUSHL IO_CHANNEL
                                01 FB 00006 CALLS #1, GET_CCB
                                50 D0 0000B MOVL R0, CCB
                                34 AB D0 0000E MOVL 52(CURRENT_VCB), MVL
                                0B A0 9A 00012 MOVZBL 11(MVL), NVOLS
                                52 24 A0 9E 00016 MOVAB 36(R0), MVL_ADDR
                                54 01 CE 0001A MNEGL #1, I
                                54 11 0001D BRB 2$
                                07 A244 7F 0001F 1$: PUSHAQ 7(MVL_ADDR)[I]
                                00 E1 00023 BBC #0, @SP)+, 2$
                                06 A244 7F 00027 PUSHAQ 6(MVL_ADDR)[I]
                                00 ED 0002B CMPZV #0, #8, @SP)+, UNIT
                                40 12 00031 BNEQ 2$
                                53 04 AC D0 00033 MOVL UNIT, R3
                                6A43 DD 00037 PUSHL (UCB_LIST)[R3]
                                7E 02 7D 0003A MOVQ #2, =(SP)
                                5E DD 0003D PUSHL SP
                                0000G CF 9F 0003F PUSHAB SEND_ERRLOG
                                05 FB 00043 CALLS #5, @SYSSCMKRN
                                07 A244 7F 0004A PUSHAQ 7(MVL_ADDR)[I]
                                9E 01 8A 0004E BICB2 #1, @SP)+
                                66 6A43 D0 00051 MOVL (UCB_LIST)[R3], (CCB)
                                7E 7C 00055 CLRQ -(SP)
                                7E 7C 00057 CLRQ -(SP)

                                : 1272
                                : 1322
                                : 1323
                                : 1324
                                : 1325
                                : 1330
                                : 1333
                                : 1337
                                : 1339
                                : 1343
                                : 1344
```


MOUVOL
V04-000

I 11
16-Sep-1984 02:25:33
14-Sep-1984 12:46:44

VAX-11 Bliss-32 V4.0-742
[MTAACP.SRC]MOUVOL.B32;1

Page 25
(6)

			7E	7C	00059	CLRQ	-(SP)	
			7E	7C	0005B	CLRQ	-(SP)	
			7E	D4	0005D	CLRL	-(SP)	
	7E	0281	8F	3C	0005F	MOVZWL	#641, -(SP)	1345
		0000G	CF	DD	00064	PUSHL	IO CHANNEL	1344
			7E	D4	00068	CLRL	-(SP)	
	00000000G	9F	0C	FB	0006A	CALLS	#12, @#SYSSQIOW	
			04	11	00071	BRB	3\$	1335
AB		54	55	F2	00073	AOBLSS	NVOLS, I, 1\$	1327
		7E	AC	7D	00077	MOVQ	UNIT, -(SP)	1353
	0000V	CF	02	FB	0007B	CALLS	#2, MAKE_CUR_VOL	
			04	00080	RET			1354

; Routine Size: 129 bytes, Routine Base: \$CODE\$ + 036E

```

847 1355 1 ROUTINE MAKE_CUR_VOL (UNIT, VOL) : NOVALUE MVL_UCB =
848 1356 1
849 1357 1 ++
850 1358 1
851 1359 1 FUNCTIONAL DESCRIPTION:
852 1360 1 This routine makes the relative volume number on the rel unit the
853 1361 1 current volume it notes that the current volume is mounted on this
854 1362 1 relative unit
855 1363 1
856 1364 1 CALLING SEQUENCE:
857 1365 1 MAKE_CUR_VOL(ARG1,ARG2) in kernel mode
858 1366 1
859 1367 1 INPUT PARAMETERS:
860 1368 1 ARG1 - relative unit number on which the relative volume is mounted
861 1369 1 ARG2 - relative volume number(starts at 1)
862 1370 1
863 1371 1 IMPLICIT INPUTS:
864 1372 1 NONE
865 1373 1
866 1374 1 OUTPUT PARAMETERS:
867 1375 1 NONE
868 1376 1
869 1377 1 IMPLICIT OUTPUTS:
870 1378 1 NONE
871 1379 1
872 1380 1 ROUTINE VALUE:
873 1381 1 NONE
874 1382 1
875 1383 1 SIDE EFFECTS:
876 1384 1 NONE
877 1385 1
878 1386 1 --
879 1387 1
880 1388 2 BEGIN
881 1389 2
882 1390 2 EXTERNAL REGISTER
883 1391 2 MVL_ENTRY = 9 : REF BBLOCK, ! address of current rel vol in mvl
884 1392 2 UCB_LIST = 10 : REF VECTOR, ! address of ucb's allocated
885 1393 2 COMMON_REG;
886 1394 2
887 1395 2 LOCAL
888 1396 2 CCB : REF BBLOCK; ! address of channel control block
889 1397 2
890 1398 2 ! calculate address of channel control block
891 1399 2
892 1400 2 CCB = GET_CCB ( .IO_CHANNEL );
893 1401 2
894 1402 2 ! assign channel to unit and set current ucb
895 1403 2
896 1404 2 CCB[CCB$L_UCB] = .UCB_LIST[UNIT];
897 1405 2 CURRENT_UCB = .UCB_LIST[UNIT];
898 1406 2
899 1407 2 ! now set volume control block fields
900 1408 2
901 1409 2 CURRENT_VCB[VCB$W_RVN] = .UNIT;
902 1410 2 CURRENT_VCB[VCB$B_CUR_RVN] = .VOL;
903 1411 2
```



```

: 904      1412  2      : note which unit the volume is mounted on
: 905      1413  2
: 906      1414  2      MVL_ENTRY [ MVL$B_RVN ] = .UNIT;
: 907      1415  1      END;
                                ! end of routine MAKE_CUR_VOL

```

			0000 00000 MAKE_CUR_VOL:			
				WORD	Save nothing	: 1355
				PUSHL	IO_CHANNEL	: 1400
				CALLS	#1, GET_CCB	
				MOVL	UNIT, RT	: 1404
				MOVL	(UCB_LIST)[R1], (CCB)	
				MOVL	(UCB_LIST)[R1], CURRENT_UCB	: 1405
				MOVW	R1, T4(CURRENT_VCB)	: 1409
				MOVB	VOL, 47(CURRENT_VCB)	: 1410
				MOVB	R1, 6(MVL_ENTRY)	: 1414
				RET		: 1415

; Routine Size: 39 bytes, Routine Base: \$CODE\$ + 03EF

```

909 1416 1 ROUTINE MAKE_VOL_ENTRY (VOL, MVL) : COMMON_CALL =
910 1417 1
911 1418 1 ++
912 1419 1
913 1420 1 FUNCTIONAL DESCRIPTION:
914 1421 1 This routine puts a relative volume in the magnetic tape volume
915 1422 1 list by making a new block and deallocating the old one
916 1423 1
917 1424 1 CALLING SEQUENCE:
918 1425 1 MAKE_VOL_ENTRY(ARG1,ARG2), CALLED IN KERNEL MODE
919 1426 1
920 1427 1 INPUT PARAMETERS:
921 1428 1 ARG1 - volume number
922 1429 1 ARG2 - address of magnetic tape volume list
923 1430 1
924 1431 1 IMPLICIT INPUTS:
925 1432 1 CURRENT_VCB - address of current volume control block
926 1433 1
927 1434 1 OUTPUT PARAMETERS:
928 1435 1 NONE
929 1436 1
930 1437 1 IMPLICIT OUTPUTS:
931 1438 1 MVL rebuilt
932 1439 1
933 1440 1 ROUTINE VALUE:
934 1441 1 address of mvl control block
935 1442 1
936 1443 1 SIDE EFFECTS:
937 1444 1 NONE
938 1445 1
939 1446 1 --
940 1447 1
941 1448 2 BEGIN
942 1449 2
943 1450 2 EXTERNAL REGISTER
944 1451 2 COMMON_REG;
945 1452 2
946 1453 2 MAP
947 1454 2 MVL : REF BBLOCK; ! addr of mag tape volume list
948 1455 2
949 1456 2 LOCAL
950 1457 2
951 1458 2 MVL_ADDR: REF BBLOCKVECTOR [10, MVL$K_LENGTH],
952 1459 2 ! address of MVL control block
953 1460 2 NEWMVL : REF BBLOCK, ! address of new MVL
954 1461 2 NVOL; ! number of volumes
955 1462 2
956 1463 2 EXTERNAL ROUTINE
957 1464 2 ALLOCATE, ! allocate non_paged system mem
958 1465 2 DEALLOCATE; ! deallocate non_paged sys mem
959 1466 2
960 1467 2 VOL = .VOL + 4; ! grab some extra slots
961 1468 2 NVOL = .MVL[MVL$B NVOLS]; ! get # of vols it will hold
962 1469 2 NEWMVL = ALLOCATE((.VOL*MVL$K_LENGTH) + MVL$K_FIXLEN); ! allocate non_paged sys space
963 1470 2
964 1471 2
965 1472 2 ! initialize new MVL
```



```
! end of routine MAKE_VOL_ENTRY
```

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

; Routine Size: 112 bytes, Routine Base: \$CODES + 0416


```

: 997 1503 1 ROUTINE CREATE_LABEL (VOL, MVL) : COMMON_CALL =
: 998 1504 1
: 999 1505 1 ++
: 1000 1506 1
: 1001 1507 1 FUNCTIONAL DESCRIPTION:
: 1002 1508 1 This routine will create a label for a given volume if it does not
: 1003 1509 1 already have a label. It does this by getting the first four characters
: 1004 1510 1 of the label from the MVL of the previous reel in the volume set;
: 1005 1511 1 padding the blanks with the underscore character and putting in the
: 1006 1512 1 RVN of the current reel as the last two chars. This means
: 1007 1513 1 that only up to 99 different labels can be generate for a given volume
: 1008 1514 1 set.
: 1009 1515 1
: 1010 1516 1 CALLING SEQUENCE:
: 1011 1517 1 CREATE_LABEL(ARG1,ARG2), CALLED IN KERNEL MODE
: 1012 1518 1
: 1013 1519 1 INPUT PARAMETERS:
: 1014 1520 1 ARG1 - volume number
: 1015 1521 1 ARG2 - address of magnetic tape volume list
: 1016 1522 1
: 1017 1523 1 IMPLICIT INPUTS:
: 1018 1524 1 CURRENT_VCB - address of current volume control block
: 1019 1525 1
: 1020 1526 1 OUTPUT PARAMETERS:
: 1021 1527 1 NONE
: 1022 1528 1
: 1023 1529 1 IMPLICIT OUTPUTS:
: 1024 1530 1 MVL for the current reel is given a Volume Label
: 1025 1531 1
: 1026 1532 1 ROUTINE VALUE:
: 1027 1533 1 True : The label could be generated
: 1028 1534 1 False : The label could not be generated
: 1029 1535 1
: 1030 1536 1 SIDE EFFECTS:
: 1031 1537 1 NONE
: 1032 1538 1
: 1033 1539 1 --
: 1034 1540 1
: 1035 1541 1 BEGIN
: 1036 1542 2
: 1037 1543 2 EXTERNAL REGISTER
: 1038 1544 2 COMMON_REG;
: 1039 1545 2
: 1040 1546 2 MAP
: 1041 1547 2 MVL : REF BBLOCK; ! addr of mag tape volume list
: 1042 1548 2
: 1043 1549 2 LOCAL
: 1044 1550 2
: 1045 1551 2 CUR_MVL_ADDR: REF BBLOCK, ! address of MVL control block
: 1046 1552 2 PRE_MVL_ADDR: REF BBLOCK, ! address of MVL control block
: 1047 1553 2 LABEL_ADDR : REF VECTOR [,BYTE], ! Place in label top reel numb
: 1048 1554 2 NUMBER_OF_TAPE, ! Number of current reel mod 99
: 1049 1555 2 DESCR : VECTOR [2]; ! Descr for FAO string
: 1050 1556 2
: 1051 1557 2
: 1052 1558 2
: 1053 1559 2
```

```
: 1054      1560 2 ! Get the number of the previous volume
: 1055      1561 2 ! and if that volume number is less than 0 then the current volume we
: 1056      1562 2 ! are using must be the first volume in the reel
: 1057      1563 2 !
: 1058      1564 2 !     IF (.VOL-1) LEQ 0 THEN RETURN FALSE;
: 1059      1565 2 !
: 1060      1566 2 ! Get the individual MVL portion of the MVL block
: 1061      1567 2 ! and move the first four chars of the previous reel's label into
: 1062      1568 2 ! the current reel's label
: 1063      1569 2 !
: 1064      1570 2 !     PRE_MVL_ADDR = .MVL + MVL$K_FIXLEN + (((.VOL-1) - 1) * MVL$K_LENGTH);
: 1065      1571 2 !     CUR_MVL_ADDR = .MVL + MVL$K_FIXLEN + ((.VOL-1) * MVL$K_LENGTH);
: 1066      1572 2 !     CHSMOVE((MVL$$VOLLBL-2), PRE_MVL_ADDR [ MVL$T_VOLLBL ],
: 1067      1573 2 !         CUR_MVL_ADDR [ MVL$T_VOLLBL ]);
: 1068      1574 2 !
: 1069      1575 2 ! Check thru the new label for blanks and overwrite them with the underscore
: 1070      1576 2 ! character.
: 1071      1577 2 !
: 1072      1578 2 !     LABEL_ADDR = CUR_MVL_ADDR [ MVL$T_VOLLBL ];
: 1073      1579 2 !     DECR I FROM MVL$$VOLLBL TO 2 DO
: 1074      1580 2 !         IF .LABEL_ADDR[.I] EQL ' '
: 1075      1581 2 !             THEN LABEL_ADDR[.I] = '%C'_;
: 1076      1582 2 !
: 1077      1583 2 ! Now check to see if the RVN of this reel is greater than 99 decimal. If
: 1078      1584 2 ! it is then set it to the RVN modulo 99.
: 1079      1585 2 !
: 1080      1586 2 !     NUMBER_OF_TAPE = .VOL;
: 1081      1587 2 !     IF .NUMBER_OF_TAPE GTR 99
: 1082      1588 2 !         THEN NUMBER_OF_TAPE = (.NUMBER_OF_TAPE MOD 99);
: 1083      1589 2 !
: 1084      1590 2 ! Set up the descriptors for the call to FAO and call FAO to convert the number
: 1085      1591 2 ! to an ASCII string and insert it into the label field in the MVL of the
: 1086      1592 2 ! current volume. Then set that the MVL is used and return to caller.
: 1087      1593 2 !
: 1088      1594 2 !     DESCR[0] = 2; ! Size of output buffer
: 1089      1595 2 !     DESCR[1] = CUR_MVL_ADDR [ MVL$T_VOLLBL ] + (MVL$$VOLLBL-2); ! Addr of output buffer
: 1090      1596 2 !
: 1091      1597 2 !     SYSSFAO (DESCRIPTOR ('!2ZB'),
: 1092      1598 2 !         0,
: 1093      1599 2 !         DESCR,
: 1094      1600 2 !         .NUMBER_OF_TAPE);
: 1095      1601 2 !     CUR_MVL_ADDR[ MVL$V_UNUSED ] = 0;
: 1096      1602 2 !     RETURN TRUE;
: 1097      1603 1 ! END;
```

```
42 5A 32 21 00486 P.AAD: .ASCII \!2ZB\
0048A .BLKB 2
00000004 0048C P.AAC: .LONG 4
00000000 00490 .ADDRESS P.AAD
```

```
0004 00000 CREATE_LABEL:
5E 08 C2 00002 .WORD Save R2
SUBL2 #8, SP
```

```
: 1503
:
```


		01	04	AC	D1	00005	CMPL	VOL, #1	:	1564	
				6B	15	00009	BLEQ	4\$:		
		50	04	AC	D0	0000B	MOVL	VOL, R0	:	1570	
		51	08	BC40	7E	0000F	MOVAQ	@MVL[R0], PRE_MVL_ADDR	:		
		51		14	C0	00014	ADDL2	#20, PRE_MVL_ADDR	:		
		52	08	BC40	7E	00017	MOVAQ	@MVL[R0], CUR_MVL_ADDR	:	1571	
		52		1C	C0	0001C	ADDL2	#28, CUR_MVL_ADDR	:		
		62		61	D0	0001F	MOVL	(PRE_MVL_ADDR), (CUR_MVL_ADDR)	:	1573	
		50		52	D0	00022	MOVL	CUR_MVL_ADDR, LABEL_ADDR	:	1578	
		51		06	D0	00025	MOVL	#6, I	:	1580	
		20		6140	91	00028	1\$: CMPB	(I)[LABEL_ADDR], #32	:		
				05	12	0002C	BNEQ	2\$:		
		51	5F	8F	90	0002E	MOVB	#95, (I)[LABEL_ADDR]	:	1581	
FFEE				8F	02	9D	00033	2\$: ACBB	#2, #-1, I, 1\$	1580	
		50	04	AC	D0	0003A	MOVL	VOL, NUMBER_OF_TAPE	:	1586	
		00000063		8F	50	D1	0003E	CMPL	NUMBER_OF_TAPE, #99	1587	
					0E	15	00045	BLEQ	3\$		
7E	00	50		50	01	7A	00047	EMUL	#1, NUMBER_OF_TAPE, #0, -(SP)	1588	
50	50			8E	00000063	8F	7B	0004C	EDIV	#99, (SP)+, NUMBER_OF_TAPE, NUMBER_OF_TAPE	
				6E	02	D0	00055	3\$: MOVL	#2, DESCR	1594	
		04	AE	04	A2	9E	00058	MOVAB	4(CUR_MVL_ADDR), DESCR+4	1595	
					50	DD	0005D	PUSHL	NUMBER_OF_TAPE	1600	
				04	AE	9F	0005F	PUSHAB	DESCR	1597	
					7E	D4	00062	CLRL	-(SP)		
				91	AF	9F	00064	PUSHAB	P.AAC		
	00000000G	9F		04	FB	00067	CALLS	#4, @#SYSS\$FAO	:		
	07	A2		02	8A	0006E	BICB2	#2, 7(CUR_MVL_ADDR)	:	1601	
		50		01	D0	00072	MOVL	#1, R0	:	1602	
					04	00075	RET		:		
				50	D4	00076	4\$: CLRL	R0	:	1603	
					04	00078	RET		:		

; Routine Size: 121 bytes, Routine Base: \$CODE\$ + 0494

; 1098 1604 1
; 1099 1605 1

```
: 1101      1606 1 ROUTINE ASSUME_MOUNTED : NOVALUE MVL_UCB =
: 1102      1607 1
: 1103      1608 1 ++
: 1104      1609 1
: 1105      1610 1 FUNCTIONAL DESCRIPTION:
: 1106      1611 1     This routine indicates that the volume is mounted
: 1107      1612 1     and sets position pointers to the beginning of tape
: 1108      1613 1
: 1109      1614 1 CALLING SEQUENCE:
: 1110      1615 1     ASSUME_MOUNTED(ARG1), CALLED IN KERNEL MODE
: 1111      1616 1
: 1112      1617 1 INPUT PARAMETERS:
: 1113      1618 1     NONE
: 1114      1619 1
: 1115      1620 1 IMPLICIT INPUTS:
: 1116      1621 1     MVL_ENTRY - address of current rel volume entry in mvl
: 1117      1622 1     CURRENT_VCB - address of current volume control block
: 1118      1623 1
: 1119      1624 1 OUTPUT PARAMETERS:
: 1120      1625 1     NONE
: 1121      1626 1
: 1122      1627 1 IMPLICIT OUTPUTS:
: 1123      1628 1     NONE
: 1124      1629 1
: 1125      1630 1 ROUTINE VALUE:
: 1126      1631 1     NONE
: 1127      1632 1
: 1128      1633 1 SIDE EFFECTS:
: 1129      1634 1     NONE
: 1130      1635 1
: 1131      1636 1 --
: 1132      1637 1
: 1133      1638 2 BEGIN
: 1134      1639 2
: 1135      1640 2 EXTERNAL REGISTER
: 1136      1641 2     MVL_ENTRY = 9 : REF BBLOCK,
: 1137      1642 2     COMMON_REG;
: 1138      1643 2
: 1139      1644 2 MVL_ENTRY [ MVL$V_MOUNTED ] = 1;           ! set it mounted
: 1140      1645 2 CURRENT_VCB[VCB$B_TM] = 0;
: 1141      1646 2 CURRENT_VCB[VCB$L_ST_RECORD] = 0;
: 1142      1647 2 CURRENT_VCB[VCB$V_LOGICEOVS] = 0;
: 1143      1648 1 END;                                     ! end of routine ASSUMED_MOUNTED
```

```
0000 00000 ASSUME_MOUNTED:
07 A9      01 88 00002      .WORD      Save nothing      : 1606
                2E AB 94 00006      BISB2      #1, 7(MVL_ENTRY) : 1644
0B AB      02 8A 0000C      CLRB      46(CURRENT_VCB)      : 1645
                30 AB D4 00009      CLRL      48(CURRENT_VCB)      : 1646
                04 00010      BICB2      #2, 11(CURRENT_VCB) : 1647
                                RET                                : 1648
```

; Routine Size: 17 bytes, Routine Base: \$CODE\$ + 050D

MOUVOL
V04-000

F 12
16-Sep-1984 02:25:33
14-Sep-1984 12:46:44

VAX-11 Bliss-32 V4.0-742
[MTAACP.SRC]MOUVOL.B32;1

Page 35
(10)

MOU
V04

; R

```
: 1145      1649 1 ROUTINE OPERATOR_LBL : MVL_UCB =
: 1146      1650 1
: 1147      1651 1 !++
: 1148      1652 1
: 1149      1653 1 FUNCTIONAL DESCRIPTION:
: 1150      1654 1     This routine records the label entered by the operator
: 1151      1655 1
: 1152      1656 1 CALLING SEQUENCE:
: 1153      1657 1     OPERATOR_LBL, CALLED IN KERNEL MODE
: 1154      1658 1
: 1155      1659 1 INPUT PARAMETERS:
: 1156      1660 1     NONE
: 1157      1661 1
: 1158      1662 1 IMPLICIT INPUTS:
: 1159      1663 1     operator mailbox message and size
: 1160      1664 1
: 1161      1665 1 OUTPUT PARAMETERS:
: 1162      1666 1     NONE
: 1163      1667 1
: 1164      1668 1 IMPLICIT OUTPUTS:
: 1165      1669 1     Magnetic Tape volume label
: 1166      1670 1
: 1167      1671 1 ROUTINE VALUE:
: 1168      1672 1     Value of error number
: 1169      1673 1
: 1170      1674 1 SIDE EFFECTS:
: 1171      1675 1     NONE
: 1172      1676 1
: 1173      1677 1 USER ERRORS:
: 1174      1678 1     NONE
: 1175      1679 1
: 1176      1680 1 --
: 1177      1681 1
: 1178      1682 2 BEGIN
: 1179      1683 2
: 1180      1684 2 EXTERNAL REGISTER
: 1181      1685 2     MVL_ENTRY = 9 : REF BBLOCK,
: 1182      1686 2     COMMON_REG;
: 1183      1687 2
: 1184      1688 2 EXTERNAL                                ! translation table for ANSI 'a' char
: 1185      1689 2     ANSI_A_BAD : VECTOR [ , BYTE ],
: 1186      1690 2     ESC_CHAR : BYTE;
: 1187      1691 2
: 1188      1692 2 LOCAL
: 1189      1693 2     OPR_INPUT : REF VECTOR [ , BYTE],
: 1190      1694 2     TEMP_LABEL : VECTOR [ MVL$S_VOLLBL, BYTE ];
: 1191      1695 2
: 1192      1696 2
: 1193      1697 2 OPR_INPUT = MAIL[OPC$L_MS_TEXT];
: 1194      1698 2
: 1195      1699 2 ! check length of label for volume
: 1196      1700 2
: 1197      1701 2 IF .MAILSZ GTRU MVL$S_VOLLBL
: 1198      1702 2 THEN RETURN INITS_MTLBLLONG
: 1199      1703 2 ELSE
: 1200      1704 2     BEGIN
: 1201      1705 2
```



```
: 1202      1706      3      ! translate the label into upper case and test for invalid characters
: 1203      1707      3
: 1204      1708      3      !
: 1205      1709      3      IF 0 NEQ MOVTUC ( MAILSZ, MAIL[OPCSL_MS_TEXT], ESC_CHAR, ANSI_A_BAD,
: 1206      1710      3      XREF ( MVL$S_VOLLBL ); TEMP_LABEL )
: 1207      1711      3      THEN RETURN INIT$ _MTLBLNONA
: 1208      1712      4      ELSE
: 1209      1713      4      BEGIN
: 1210      1714      4      CH$COPY ( .MAILSZ, TEMP_LABEL, ' '
: 1211      1715      4      MVL$S_VOLLBL, MVL_ENTRY [ MVL$T_VOLLBL ] );
: 1212      1716      4      MVL_ENTRY [ MVL$V_UNUSED ] = 0;
: 1213      1717      3      RETURN 1;
: 1214      1718      3      END;
: 1215      1719      1      END;

! end of routine OPERATOR_LBL
```

.EXTRN ANSI_A_BAD, ESC_CHAR

007C 00000 OPERATOR_LBL:

	56	0000G	CF	9E	00002	WORD	Save R2,R3,R4,R5,R6	1649
	5E		08	C2	00007	MOVAB	MAILSZ, R6	
	50	8C	A6	9E	0000A	SUBL2	#8, SP	1697
	06		66	D1	0000E	MOVAB	MAIL+8, OPR_INPUT	1701
			08	1B	00011	CMPL	MAILSZ, #6	
	50	00758104	8F	D0	00013	BLEQU	1\$	1704
				04	0001A	MOVL	#7700740, R0	
0000G	CF	0000G	CF	8C	A6	RET		
			66	2F	0001B	MOVTUC	MAILSZ, MAIL+8, ESC_CHAR, ANSI_A_BAD, #6, -	1708
			06		00025		TEMP_LABEL	
			02	1D	00027	BVS	2\$	
			51	D4	00029	CLRL	R1	
			51	D5	0002B	TSTL	R1	
			08	13	0002D	BEQL	3\$	
	50	0075810C	8F	D0	0002F	MOVL	#7700748, R0	1712
				04	00036	RET		
06		20	6E	66	2C	MOVCS	MAILSZ, TEMP_LABEL, #32, #6, (MVL_ENTRY)	1714
				69				
	07	A9	02	8A	0003D	BICB2	#2, 7(MVL_ENTRY)	1715
		50	01	D0	00041	MOVL	#1, R0	1716
				04	00044	RET		1719

; Routine Size: 69 bytes, Routine Base: \$CODE\$ + 051E

```
1217 1720 1 ROUTINE CHECK_RING : COMMON_CALL =
1218 1721 1
1219 1722 1 ++
1220 1723 1
1221 1724 1 FUNCTIONAL DESCRIPTION:
1222 1725 1 Check to see if the write ring is in the tape.
1223 1726 1
1224 1727 1 CALLING SEQUENCE:
1225 1728 1 CHECK_RING ( )
1226 1729 1
1227 1730 1 INPUT PARAMETERS:
1228 1731 1 none
1229 1732 1
1230 1733 1 IMPLICIT INPUTS:
1231 1734 1 CURRENT_VCB - address of current volume control block
1232 1735 1 IO_CHANNEL - ACP IO channel
1233 1736 1
1234 1737 1 OUTPUT PARAMETERS:
1235 1738 1 none
1236 1739 1
1237 1740 1 IMPLICIT OUTPUTS:
1238 1741 1 none
1239 1742 1
1240 1743 1 ROUTINE VALUE:
1241 1744 1 0 - the volume is Hardware write lock ( NO RING )
1242 1745 1 1 - the volume is NOT Hardware write lock ( RING )
1243 1746 1
1244 1747 1 SIDE EFFECTS:
1245 1748 1 none
1246 1749 1
1247 1750 1 --
1248 1751 1
1249 1752 2 BEGIN
1250 1753 2
1251 1754 2 EXTERNAL REGISTER
1252 1755 2 COMMON_REG;
1253 1756 2
1254 1757 2 EXTERNAL ROUTINE
1255 1758 2 ISSUE_IO : L$ISSUE_IO;
1256 1759 2
1257 1760 2 EXTERNAL
1258 1761 2 IO_STATUS : VECTOR [ 2, LONG ], ! QIO's Status
1259 1762 2 USER_STATUS : VECTOR [ 2, LONG ]; ! User's Status
1260 1763 2
1261 1764 2 LOCAL
1262 1765 2 STATUS : LONG; ! status of IO
1263 1766 2
1264 1767 2 ! get at the information nicely
1265 1768 2
1266 1769 2 BIND DEVICE_DEPENDENT = IO_STATUS [ 1 ] : BBLOCK;
1267 1770 2
1268 1771 2
1269 1772 2 ! do a sensemode operation
1270 1773 2
1271 1774 2 STATUS = ISSUE_IO ( IO$_SENSEMODE, 0, 0 );
1272 1775 2
1273 1776 2 IF NOT .STATUS
```


MOUVOL
V04-000

J 12
16-Sep-1984 02:25:33
14-Sep-1984 12:46:44

VAX-11 Bliss-32 V4.0-742
[MTAACP.SRC]MOUVOL.B32;1

Page 39
(12)

: 1274 1777 2
: 1275 1778 3
: 1276 1779 3
: 1277 1780 3
: 1278 1781 3
: 1279 1782 2
: 1280 1783 2
: 1281 1784 2
: 1282 1785 1

```
THEN
  BEGIN
    USER_STATUS[0] = .STATUS;
    USER_STATUS[1] = $$$_FCPREADERR;
    ERR_EXIT();
  END;

RETURN NOT (.DEVICE_DEPENDENT [ MTSV_HWL ]);
END;                                ! end of routine CHECK_RING
```

.EXTRN ISSUE_IO, USER_STATUS

0000 00000 CHECK_RING:

			7E 7C 00002	.WORD	Save nothing	: 1720
			27 DD 00004	CLRQ	-(SP)	: 1774
			0000G 30 00006	PUSHL	#39	
			0C C0 00009	BSBW	ISSUE_IO	
	5E		50 E8 0000C	ADDL2	#12, SP	
	OE		50 D0 0000F	BLBS	STATUS, 1\$: 1776
0000G	CF		8F 3C 00014	MOVL	STATUS, USER_STATUS	: 1779
0000G	CF	0888	00 BF 0001B	MOVZWL	#2184, USER_STATUS+4	: 1780
			03 EF 0001D	CHMU	#0	: 1781
50	0000G	CF	50 D2 00024	EXTZV	#3, #1, DEVICE_DEPENDENT+2, R0	: 1784
			04 00027	MCOML	R0, R0	: 1785
				RET		

; Routine Size: 40 bytes, Routine Base: \$CODE\$ + 0563

: 1283 1786 1

NXT

```
1285 1787 1 ROUTINE CHECK_ACCESS ( FLAGS ): MVL_UCB =
1286 1788 1
1287 1789 1 ++
1288 1790 1
1289 1791 1 FUNCTIONAL DESCRIPTION:
1290 1792 1     this routine check the user's access rights to a tape reel
1291 1793 1
1292 1794 1 CALLING SEQUENCE:
1293 1795 1     CHECK_ACCESS ( ARG1 )
1294 1796 1
1295 1797 1 INPUT PARAMETERS:
1296 1798 1     ARG1 - the mount_volume flages ( passed by value )
1297 1799 1
1298 1800 1 IMPLICIT INPUTS:
1299 1801 1     This routine assumes that the operator has responed to the MTAACP via
1300 1802 1     the REPLY command and that MAIL[OPC$W_MS_STATUS] is set to some value
1301 1803 1     according to that responce.
1302 1804 1
1303 1805 1 OUTPUT PARAMETERS:
1304 1806 1     NONE
1305 1807 1
1306 1808 1 IMPLICIT OUTPUTS:
1307 1809 1     MVL_ENTRY [ MVL$V_OVERRIDE ] is set correctly
1308 1810 1
1309 1811 1 ROUTINE VALUE:
1310 1812 1     TRUE - if the uses has the needed rights to access the tape
1311 1813 1     FALSE - otherwise
1312 1814 1
1313 1815 1 SIDE EFFECTS:
1314 1816 1     NONE
1315 1817 1
1316 1818 1 USER ERRORS:
1317 1819 1     NONE
1318 1820 1
1319 1821 1 --
1320 1822 1
1321 1823 2 BEGIN
1322 1824 2
1323 1825 2 MAP
1324 1826 2     FLAGS          : BBLOCK;
1325 1827 2
1326 1828 2 EXTERNAL REGISTER
1327 1829 2     MVL_ENTRY = 9   : REF BBLOCK,    ! addr of MVL entry for current vol
1328 1830 2     COMMON_REG;
1329 1831 2
1330 1832 2 EXTERNAL
1331 1833 2     ANSI_A_GOOD    : VECTOR [ , BYTE ]; ! translation table for ANSI 'a' char
1332 1834 2
1333 1835 2 LOCAL
1334 1836 2     ACCESS,          ! Access to tape from $MTACCESS
1335 1837 2     CURRENT_RECORD,  ! current record tape drive is reading
1336 1838 2     DEVCHAR          : REF BBLOCK,
1337 1839 2     TEMP_LABEL       : VECTOR [ VL1$S_VOLLBL, BYTE ],
1338 1840 2     TAPE_OWNER_STS   : LONG,
1339 1841 2     VMS_TAPE,
1340 1842 2     STATUS           : LONG,
1341 1843 2     ORB              : REF BBLOCK,    ! ORB address
```



```

: 1342      1844      2
: 1343      1845      2
: 1344      1846      2
: 1345      1847      2
: 1346      1848      2
: 1347      1849      2
: 1348      1850      2
: 1349      1851      2
: 1350      1852      2
: 1351      1853      2
: 1352      1854      2
: 1353      1855      2
: 1354      1856      2
: 1355      1857      2
: 1356      1858      2
: 1357      1859      2
: 1358      1860      2
: 1359      1861      2
: 1360      1862      2
: 1361      1863      2
: 1362      1864      2
: 1363      1865      2
: 1364      1866      2
: 1365      1867      2
: 1366      1868      2
: 1367      1869      2
: 1368      1870      2
: 1369      1871      2
: 1370      1872      3
: 1371      1873      2
: 1372      1874      3
: 1373      1875      3
: 1374      1876      3
: 1375      1877      3
: 1376      1878      5
: 1377      1879      4
: 1378      1880      5
: 1379      1881      4
: 1380      1882      3
: 1381      1883      4
: 1382      1884      4
: 1383      1885      4
: 1384      1886      4
: 1385      1887      4
: 1386      1888      4
: 1387      1889      4
: 1388      1890      3
: 1389      1891      4
: 1390      1892      4
: 1391      1893      4
: 1392      1894      4
: 1393      1895      3
: 1394      1896      2
: 1395      1897      2
: 1396      1898      2
: 1397      1899      2
: 1398      1900      2

      MVL          : REF BBLOCK,
      SCRATCH      : REF BBLOCK,
      WRITE_RING   : BITVECTOR [1];

EXTERNAL
HDR1              : REF BBLOCK;    ! address of HDR1(E0F1) label

! get a handle on the current MVL
MVL = .CURRENT_VCB[VCB$MVL];

! setup address of scratch area and read the VOL1 label
SCRATCH = .HDR1 + SCRATCH_OFFSET;

! setup default tape owner and protection
ORB = .CURRENT_UCB[UCB$ORB];
TAPE_OWNER = .ORB[ORB$OWNER];
TAPE_PROT = 0;

! read the first block on the tape
STATUS = READ_BLOCK(.SCRATCH, ANSI_LBLSZ);

! if this is ( a NOT valid ANSI tape)
IF NOT ( .STATUS AND (.SCRATCH[VL1$L_VL1LID] EQL 'VOL1'))
THEN
  BEGIN
    ! AND a valid init THEN RETURN
    IF (      (.MAIL[OPC$W_MS_STATUS] EQL (OPC$_INITAPE AND %X'FFFF'))
        OR   .CURRENT_VCB[VCB$V_INIT]
        AND (NOT (.FLAGS[MOU$V_LBLCHECK] OR .FLAGS[MOU$V_CHKIFSPC]))
      )
    THEN
      BEGIN
        KERNEL_CALL ( SET_MVL_OVERRIDE, TRUE );
        RETURN TRUE;
      END
    ! else it is an error
  ELSE
    BEGIN
      PRINT_OPR_MSG(MOUN$ NOTANSI, 0,
                    .CVT_DEVNAM_LENGTH, CVT_DEVNAM);
      RETURN FALSE;
    END;
  END;

! Set the override switch in this volumes portion of the MVL.
! If the user has specified override privilege in the MOUNT command.
```

M 12
16-Sep-1984 02:25:33 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:46:44 [MTAACP.SRC]MOUVOL.B32;1

**NXT
V04**

```

1399 1901 222
1400 1902 222
1401 1903 222
1402 1904 222
1403 1905 222
1404 1906 222
1405 1907 222
1406 1908 222
1407 1909 222
1408 1910 222
1409 1911 222
1410 1912 222
1411 1913 222
1412 1914 222
1413 1915 222
1414 1916 222
1415 1917 222
1416 1918 222
1417 1919 222
1418 1920 222
1419 1921 222
1420 1922 222
1421 1923 222
1422 1924 222
1423 1925 222
1424 1926 222
1425 1927 222
1426 1928 222
1427 1929 222
1428 1930 222
1429 1931 222
1430 1932 222
1431 1933 222
1432 1934 222
1433 1935 222
1434 1936 222
1435 1937 222
1436 1938 222
1437 1939 222
1438 1940 222
1439 1941 222
1440 1942 222
1441 1943 222
1442 1944 222
1443 1945 222
1444 1946 222
1445 1947 222
1446 1948 222
1447 1949 222
1448 1950 222
1449 1951 222
1450 1952 222
1451 1953 222
1452 1954 222
1453 1955 222
1454 1956 222
1455 1957 222

P P P P P

KERNEL_CALL ( SET_MVL_OVERRIDE, .MVL [ MVL$V_OVRPRO ] );

! Call the accessibility system service to check the accessibility char
! on the VOL1 label.
! First keep the record that the UCB is reading. The accessibility
! routine can not move the tape from under us! Thus we will compare
! this to the field after the call and if the tape was moved we punt
! the operation. Then check the codes that the routine can return
! to make sure the user has access to the tape.

CURRENT_RECORD = KERNEL_CALL (GET_RECORD, .CURRENT_UCB);
ACCESS = $MTACCESS(LBLNAM = .SCRATCH,
                   UIC = .ORB[ORBSL_OWNER],
                   STD_VERSION = .MVL[MVL$B_STDVER],
                   ACCESS_CHAR = 0,
                   ACCESS_SPEC = MTASK_NOCHAR,
                   TYPE = MTASK_INVOL1);

STATUS = KERNEL_CALL(GET_RECORD, .CURRENT_UCB);
IF .CURRENT_RECORD NEQ .STATUS
  THEN
  BEGIN
    PRINT_OPR_MSG( MOUN$_TAPEPOSLOST, 0,
                  .CVT_DEVNAM_LENGTH, CVT_DEVNAM);
    RETURN FALSE;
  END;

IF .ACCESS EQL SSS_FILACCERR
  THEN
  BEGIN
    IF NOT (.CURRENT_VCB[VCB$V_OVRACC] AND .MVL_ENTRY[MVL$V_OVERRIDE])
      THEN
      BEGIN
        PRINT_OPR_MSG(MOUN$_ACCERR, 0,
                      .CVT_DEVNAM_LENGTH, CVT_DEVNAM);
        RETURN FALSE;
      END;
    ACCESS = SSS_NORMAL;
  END;

IF .ACCESS EQL SSS_NOVOLACC
  THEN
  BEGIN
    PRINT_OPR_MSG(MOUN$_NOVOLACC, 0,
                  .CVT_DEVNAM_LENGTH, CVT_DEVNAM);
    RETURN FALSE;
  END;

IF .ACCESS EQL SSS_NOFILACC
  THEN
  BEGIN
    PRINT_OPR_MSG(MOUN$_NOFILACC, 0,
                  .CVT_DEVNAM_LENGTH, CVT_DEVNAM);
    RETURN FALSE;
  END;

```



```

: 1456      1958      2      ! Find out who owns the tape. If the field is in VMS format then
: 1457      1959      2      ! treat the field like the protection field in the VOL2 label.
: 1458      1960      2
: 1459      1961      2      TAPE_OWNER_STS = (TAPE_OWN PROT ( TAPE_OWNER, TAPE_PROT,
: 1460      1962      2      .ORB[ORB$$_OWNER], .SCRATCH ));
: 1461      1963      2
: 1462      1964      2      ! Set the override switch in this volumes portion of the MVL.
: 1463      1965      2      ! If the users UIC matches the UIC of the volume or the user
: 1464      1966      2      ! has priv.
: 1465      1967      2
: 1466      P 1968      2      KERNEL_CALL ( SET_MVL_OVERRIDE,
: 1467      P 1969      2      ( (.TAPE_OWNER EQL .ORB[ORB$$_OWNER])
: 1468      1970      2      OR .MVL [MVL$$_OVRPRO ]));
: 1469      1971      2
: 1470      1972      2      ! If the ACCESS field allows to check the VMS volume protection then
: 1471      1973      2      do so using the following rule:
: 1472      1974      2      ! If the owner identifier field of the VOL1 label was not a VMS protection
: 1473      1975      2      ! and this is a pre ANSI standard version 4 tape and there was D% INFO
: 1474      1976      2      ! in the volume field then make sure that the user has the correct priv's
: 1475      1977      2      ! to mount the tape.
: 1476      1978      2      ! Else reset the VMS protection such that the user has complete access.
: 1477      1979      2
: 1478      1980      2      IF .ACCESS
: 1479      1981      2      THEN
: 1480      1982      2      BEGIN
: 1481      1983      2      IF NOT .TAPE_OWNER_STS
: 1482      1984      2      AND ( NOT ( .CURRENT_VCB[VCB$$_OVRVOLO]
: 1483      1985      2      AND .MVL_ENTRY[MVL$$_OVERRIDE]))
: 1484      1986      2      THEN
: 1485      1987      2      BEGIN
: 1486      1988      2      PRINT OPR MSG(MOUN$$_VOLOERR,0,.CVT_DEVNAM_LENGTH,CVT_DEVNAM);
: 1487      1989      2      RETURN FALSE;
: 1488      1990      2      END
: 1489      1991      2      END
: 1490      1992      2      ELSE
: 1491      1993      2      TAPE_PROT = 0;
: 1492      1994      2
: 1493      1995      2      ! Check to see if this tape was created on VMS. If it was then we will
: 1494      1996      2      ! want to process to VOL2 label if there is one.
: 1495      1997      2
: 1496      1998      2      IF CH$EQL(10,STARID,10,SCRATCH[VL1$$_SYSCODE],0)
: 1497      1999      2      THEN VMS_TAPE = 1
: 1498      2000      2      ELSE VMS_TAPE = 0;
: 1499      2001      2
: 1500      2002      2      ! Read the VOL2 label if specified and use this as the VMS protection
: 1501      2003      2      ! for the tape, unless the ACCESS is set such that the user has complete
: 1502      2004      2      ! access to the volume. After we have read the VOL2 label we want to be sure
: 1503      2005      2      ! to reposition the tape back to teh VOL1 label to continue processing
: 1504      2006      2      ! the ANSI VOL1 label.
: 1505      2007      2
: 1506      2008      2      STATUS = READ_BLOCK (.SCRATCH, ANSI_LBLSZ);
: 1507      2009      2      IF .VMS_TAPE AND .SCRATCH[VL2$$_VL2[ID]] EQL 'VOL2'
: 1508      2010      2      THEN
: 1509      2011      2      BEGIN
: 1510      2012      2      PROCESS_VOL2_LABEL(TAPE_OWNER,TAPE_PROT,
: 1511      2013      2      .ORB[ORB$$_OWNER],.SCRATCH);
: 1512      2014      2      IF NOT .ACCESS
```

```

: 1513      2015      3          THEN TAPE_PROT = 0;
: 1514      2016      2          END;
: 1515      2017      2
: 1516      2018      2          ! Set the tape back to the begining and reread the VOL1 label.
: 1517      2019      2
: 1518      2020      2          IF NOT REWIND_AND_WAIT()
: 1519      2021      2              THEN
: 1520      2022      2                  BEGIN
: 1521      2023      2                      PRINT OPR MSG(MOUN$_IOERROR,0,..CVT_DEVNAM_LENGTH,CVT_DEVNAM);
: 1522      2024      2                      RETURN FALSE;
: 1523      2025      2                  END;
: 1524      2026      2          STATUS = READ_BLOCK (.SCRATCH, ANSI_LBLSZ);
: 1525      2027      2
: 1526      2028      2
: 1527      2029      2          ! Translate the VOL1 label into upper case and put in 'a' for any non-ANSI
: 1528      2030      2          ! a characters found ( this is done for backward compatiblility )
: 1529      2031      2
: 1530      2032      2          CH$TRANSLATE ( ANSI_A_GOOD, VL1$$_VOLLBL, SCRATCH[VL1$T_VOLLBL], ' ',
: 1531      2033      2                      VL1$$_VOLLBL, TEMP_LABEL );
: 1532      2034      2
: 1533      2035      2          ! Labels spec either generated buy the MTAACP or supplied by teh oper
: 1534      2036      2          ! must match except under the following circumstances:
: 1535      2037      2              A valid 'REPLY/INIT' or 'MOUNT/INIT'
: 1536      2038      2              A valid no label need be specified
: 1537      2039      2              A valid 'MOUNT/OVER=ID'
: 1538      2040      2
: 1539      2041      2          IF
: 1540      2042      4              (
: 1541      2043      4                  (
: 1542      2044      5                      .FLAGS[MOU$V_LBLCHECK]
: 1543      2045      6                      OR .FLAGS[MOU$V_CHKIFSPC]
: 1544      2046      5                      OR ( NOT .CURRENT_VCB[VCB$V_INIT]
: 1545      2047      4                          AND (.MAIL[OPC$W_MS STATUS]
: 1546      2048      3                              NEQ (OPC$_INITAPE AND %X'FFFF')) )
: 1547      2049      4                      )
: 1548      2050      3                      AND NOT (.FLAGS[MOU$V_CHKIFSPC] AND (NOT .LABEL_SPEC [ 0 ]))
: 1549      2051      4                      AND NOT (.CURRENT_VCB[VCB$V_OVRLBL] AND .MVL_ENTRY[MVL$V_OVERRIDE])
: 1550      2052      3                      )
: 1551      2053      2                      AND CH$NEQ ( MVL$$_VOLLBL, MVL_ENTRY[MVL$T_VOLLBL],
: 1552      2054      2                          VL1$$_VOLLBL, TEMP_LABEL )
: 1553      2055      2          THEN
: 1554      2056      2              BEGIN
: 1555      2057      2                  PRINT NOT_LABEL(.MVL_ENTRY);
: 1556      2058      2                  RETURN FALSE;
: 1557      2059      2              END;
: 1558      2060      2
: 1559      2061      2          ! check VMS volume UIC protection or user has bypass, sysprv or volpro
: 1560      2062      2
: 1561      2063      2          DEVCHAR = CURRENT_UCB[UCB$$_DEVCHAR];
: 1562      2064      2          WRITE_RING [0] = .DEVCHAR <$BITPOSITION(DEV$V_SWL) ,1>;
: 1563      2065      3          IF NOT (
: 1564      2066      3              KERNEL_CALL ( CHECK_PROT, TAPE_PROT, TAPE_OWNER,
: 1565      2067      4                  .ORB[ORB$$_OWNER],
: 1566      2068      3                  WRITE_RING )
: 1567      2069      3              )
: 1568      2070      2          THEN
: 1569      2071      3              BEGIN
```



```
: 1570      2072      3      PRINT_OPR_MSG(MOUN$ NOPRIV, 0,  
: 1571      2073      3      .CVT_DEVNAM_LENGTH, CVT_DEVNAM);  
: 1572      2074      3      RETURN FALSE;  
: 1573      2075      3      END;  
: 1574      2076      3  
: 1575      2077      3      ! set override switches are valid in the MVL if a valid init  
: 1576      2078      3  
: 1577      2079      3      DEVCHAR <$BITPOSITION(DEVS$ SWL) ,1> = .WRITE RING [0] ;  
: 1578      2080      3      IF NOT (.FLAGS[MOUS$ LBLCHECK] OR .FLAGS[MOUS$ _CHKIFSPC])  
: 1579      2081      3      THEN KERNEL_CALL ( SET_MVL_OVERRIDE, TRUE );  
: 1580      2082      3  
: 1581      2083      3      RETURN TRUE;  
: 1582      2084      3      END;  
:                               ! end of routine CHECK_ACC
```

```
.EXTRN ANSI_A_GOOD, HDR1  
.EXTRN SYSS$MTACCESS
```

05FC 00000 CHECK_ACCESS:

		5A	0000'	CF	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R10	: 1787
		5E		0C	C2	00007	MOVAB	CVT_DEVNAM, R10	
		57	34	AB	D0	0000A	SUBL2	#12, SP	
54	0000G	CF	00000140	8F	C1	0000E	MOVL	52(CURRENT_VCB), MVL	: 1854
		50	0000G	CF	D0	00018	ADDL3	#320, HDR1, SCRATCH	: 1858
		56	1C	A0	D0	0001D	MOVL	CURRENT_UCB, R0	: 1862
	F0	AA		66	D0	00021	MOVL	28(R0), -ORB	
			F4	AA	94	00025	MOVL	(ORB), TAPE_OWNER	: 1863
		7E	50	8F	9A	00028	CLRW	TAPE_PROT	: 1864
				54	DD	0002C	MOVZBL	#80, -(SP)	: 1868
	0000G	CF		02	FB	0002E	PUSHL	SCRATCH	
		58		50	D0	00033	CALLS	#2, READ_BLOCK	
		09		58	E9	00036	MOVL	R0, STATUS	
314C4F56		8F		64	D1	00039	BLBC	STATUS, 1\$: 1872
				2B	13	00040	CMPL	(SCRATCH), #827084630	
	81D3	8F	0000G	CF	B1	00042	BEQL	5\$	
				03	12	00049	CMPW	MAIL+2, #33235	: 1878
				023E	31	0004B	BNEQ	3\$	
				03	E1	0004E	BRW	31\$	
0A	2D	AB		01	EC	00053	BBC	#3, 45(CURRENT_VCB), 4\$: 1879
05	04	AC		02	E1	00058	BBS	#1, FLAGS, 4\$: 1880
EE	04	AC		5A	DD	0005D	BBC	#2, FLAGS, 2\$	
				7E	10	AA	PUSHL	R10	: 1892
						7E	MOVZBL	CVT_DEVNAM_LENGTH, -(SP)	: 1893
			007280FC	8F	DD	00065	CLRL	-(SP)	: 1892
				6A	11	0006B	PUSHL	#7504124	
				01	EF	0006D	BRB	6\$	
7E	13	A7	01	01	DD	00073	EXTZV	#1, #1, 19(MVL), -(SP)	: 1901
				01	DD	00075	PUSHL	#1	
				5E	DD	00077	PUSHL	SP	
			0000V	CF	9F	00077	PUSHL	SET_MVL_OVERRIDE	
				04	FB	0007B	PUSHAB	#4, @#SYSS\$CMKRNL	
			0000G	CF	DD	00082	CALLS	CURRENT_UCB	: 1911
				01	DD	00086	PUSHL	#1	
				5E	DD	00088	PUSHL	SP	
			0000G	CF	9F	0008A	PUSHL	GET_RECORD	
				04	FB	0008E	PUSHAB	#4, @#SYSS\$CMKRNL	

52		50	D0	00095	MOVL	R0, CURRENT_RECORD	
		7E	7C	00098	CLRQ	-(SP)	1917
		7E	D4	0009A	CLRL	-(SP)	
7E	22	A7	9A	0009C	MOVZBL	34(MVL), -(SP)	
		66	DD	000A0	PUSHL	(ORB)	
		54	DD	000A2	PUSHL	SCRATCH	
00000000G	00	06	FB	000A4	CALLS	#6, SYSSMTACCESS	
	55	50	D0	000AB	MOVL	R0, ACCESS	
		CF	DD	000AE	PUSHL	CURRENT_UCB	1919
		01	DD	000B2	PUSHL	#1	
		5E	DD	000B4	PUSHL	SP	
		CF	9F	000B6	PUSHAB	GET_RECORD	
00000000G	9F	04	FB	000BA	CALLS	#4, @#SYSSCMKRN	
	58	50	D0	000C1	MOVL	R0, STATUS	
	58	52	D1	000C4	CMPL	CURRENT_RECORD, STATUS	1920
		10	13	000C7	BEQL	7\$	
		5A	DD	000C9	PUSHL	R10	1923
7E	10	AA	9A	000CB	MOVZBL	CVT_DEVNAM_LENGTH, -(SP)	1924
		7E	D4	000CF	CLRL	-(SP)	1923
		8F	DD	000D1	PUSHL	#7504500	
		56	11	000D7	BRB	12\$	
0000009C	8F	55	D1	000D9	CMPL	ACCESS, #156	1928
		1D	12	000E0	BNEQ	10\$	
05	2C	01	E1	000E2	BBC	#1, 44(CURRENT_VCB), 8\$	1931
10	07	02	E0	000E7	BBS	#2, 7(MVL_ENTRY), 9\$	
		5A	DD	000EC	PUSHL	R10	1934
		AA	9A	000EE	MOVZBL	CVT_DEVNAM_LENGTH, -(SP)	1935
7E	10	7E	D4	000F2	CLRL	-(SP)	1934
		8F	DD	000F4	PUSHL	#7504100	
		33	11	000FA	BRB	12\$	
		01	D0	000FC	MOVL	#1, ACCESS	1938
000022A4	8F	55	D1	000FF	CMPL	ACCESS, #8868	1941
		10	12	00106	BNEQ	11\$	
		5A	DD	00108	PUSHL	R10	1944
7E	10	AA	9A	0010A	MOVZBL	CVT_DEVNAM_LENGTH, -(SP)	1945
		7E	D4	0010E	CLRL	-(SP)	1944
		8F	DD	00110	PUSHL	#7504484	
		6C	11	00116	BRB	16\$	
000022AC	8F	55	D1	00118	CMPL	ACCESS, #8876	1949
		10	12	0011F	BNEQ	13\$	
		5A	DD	00121	PUSHL	R10	1952
7E	10	AA	9A	00123	MOVZBL	CVT_DEVNAM_LENGTH, -(SP)	1953
		7E	D4	00127	CLRL	-(SP)	1952
		8F	DD	00129	PUSHL	#7504492	
		53	11	0012F	BRB	16\$	
		54	DD	00131	PUSHL	SCRATCH	1962
		66	DD	00133	PUSHL	(ORB)	
		AA	9F	00135	PUSHAB	TAPE_PROT	1961
		AA	9F	00138	PUSHAB	TAPE_OWNER	
0000G	CF	04	FB	0013B	CALLS	#4, TAPE_OWN_PROT	
	52	50	D0	00140	MOVL	R0, TAPE_OWNER_STS	
		50	D4	00143	CLRL	R0	1970
66		AA	D1	00145	CMPL	TAPE_OWNER, (ORB)	
		02	12	00149	BNEQ	14\$	
		50	D6	0014B	INCL	R0	
51	13	01	01	EF	EXTZV	#1, #1, 19(MVL), R1	
		7E	51	C9	BISL3	R1, R0, -(SP)	

				0000V	01 DD 00157	PUSHL #1	
					5E DD 00159	PUSHL SP	
					CF 9F 0015B	PUSHAB SET_MVL_OVERRIDE	
		00000000G	9F		04 FB 0015F	CALLS #4, -@SYSSCMKRNL	
			1D		55 E9 00166	BLBC ACCESS, 17\$	1980
			1D		52 E8 00169	BLBS TAPE_OWNER_STS, 18\$	1983
	05	2D	AB		05 E1 0016C	BBC #5, 45(CURRENT_VCB), 15\$	1984
	13	07	A9		02 E0 00171	BBS #2, 7(MVL_ENTRY), 18\$	1985
					5A DD 00176	PUSHL R10	1988
			7E	10	AA 9A 00178	MOVZBL CVT DEVNAM_LENGTH, -(SP)	
					7E D4 0017C	CLRL -(SP)	
				0072821C	8F DC 0017E	PUSHL #7504412	
					58 11 00184	BRB 22\$	
				F4	AA B4 00186	CLRW TAPE_PROT	1993
					0A 29 00189	CMPC3 #10, STARID, 24(SCRATCH)	1998
18	A4	F8E7	CF		05 12 00190	BNEQ 19\$	
			52		01 D0 00192	MOVL #1, VMS_TAPE	1999
					02 11 00195	BRB 20\$	
					52 D4 00197	CLRL VMS_TAPE	2000
			7E	50	8F 9A 00199	MOVZBL #80, -(SP)	2008
					54 DD 0019D	PUSHL SCRATCH	
		0000G	CF		02 FB 0019F	CALLS #2, READ_BLOCK	
			58		50 D0 001A4	MOVL R0, STATUS	
			1E		52 E9 001A7	BLBC VMS_TAPE, 21\$	2009
		324C4F56	8F		64 D1 001AA	CMPL (SCRATCH), #843861846	
					15 12 001B1	BNEQ 21\$	
					54 DD 001B3	PUSHL SCRATCH	2013
					66 DD 001B5	PUSHL (ORB)	
				F4	AA 9F 001B7	PUSHAB TAPE_PROT	2012
				F0	AA 9F 001BA	PUSHAB TAPE_OWNER	
					04 FB 001BD	CALLS #4, PROCESS_VOL2_LABEL	
		0000G	CF		55 E8 001C2	BLBS ACCESS, 21\$	2014
			03		AA B4 001C5	CLRW TAPE_PROT	2015
				F4	00 FB 001C8	CALLS #0, REWIND_AND_WAIT	2020
		0000G	CF		50 E8 001CD	BLBS R0, 23\$	
			11		5A DD 001D0	PUSHL R10	2023
					7E AA 9A 001D2	MOVZBL CVT DEVNAM_LENGTH, -(SP)	
					7E D4 001D6	CLRL -(SP)	
				00728124	8F DD 001D8	PUSHL #7504164	
					0094 31 001DE	BRW 29\$	
			7E	50	8F 9A 001E1	MOVZBL #80, -(SP)	2026
					54 DD 001E5	PUSHL SCRATCH	
		0000G	CF		02 FB 001E7	CALLS #2, READ_BLOCK	
			58		50 D0 001EC	MOVL R0, STATUS	
0000G	CF				06 2E 001EF	MOVTC #6, 4(SCRATCH), #32, ANSI_A_GOOD, #6, -	2032
	20	04	A4		06 001F7	TEMP_LABEL	
		04	AE		01 E0 001FA	BBS #1, FLAGS, 24\$	2042
	13	04	AC		02 E0 001FF	BBS #2, FLAGS, 25\$	2043
	13	04	AC		03 E0 00204	BBS #3, 45(CURRENT_VCB), 28\$	2044
	2D	2D	AB		CF B1 00209	CMPL MAIL+2, #33235	2046
		81D3	8F	0000G	24 13 00210	BEQL 28\$	
					02 E1 00212	BBC #2, FLAGS, 26\$	2049
	04	04	AC		AA E9 00217	BLBC LABEL_SPEC, 28\$	
		1B		F8	02 E1 0021B	BBC #2, 45(CURRENT_VCB), 27\$	2051
	05	2C	AB		02 E0 00220	BBS #2, 7(MVL_ENTRY), 28\$	
	11	07	A9		06 29 00225	CMPC3 #6, (MVL_ENTRY), TEMP_LABEL	2053
04	AE		69		0A 13 0022A	BEQL 28\$	

50	52	0000G	CF	59	DD	0022C	PUSHL	MVL ENTRY	2057
6E	52		01	0000G	30	0022E	BSBW	PRINT NOT_LABEL	
	01		00	04	CO	00231	ADDL2	#4, SP	
				6B	11	00234	BRB	33\$	2058
				38	C1	00236	ADDL3	#56, CURRENT_UCB, DEVCHAR	2063
				19	EF	0023C	EXTZV	#25, #1, DEVCHAR, R0	2064
				50	F0	00241	INSV	R0, #0, #1, WRITE_RING	
				5E	DD	00246	PUSHL	SP	2067
				66	DD	00248	PUSHL	(ORB)	
				F0	AA	9F	PUSHAB	TAPE_OWNER	
				F4	AA	9F	PUSHAB	TAPE_PROT	
				04	DD	00250	PUSHL	#4	
				5E	DD	00252	PUSHL	SP	
				0000G	CF	9F	PUSHAB	CHECK_PROT	
					07	FB	CALLS	#7, @SYSS\$CMKRNL	
					50	E8	BLBS	R0, 30\$	
					01	E0	BBS	#1, 19(MVL), 30\$	2068
					5A	DD	PUSHL	R10	2072
					AA	9A	MOVZBL	CVT_DEVNAM_LENGTH, -(SP)	2073
					7E	D4	CLRL	-(SP)	2072
					8F	DD	PUSHL	#7504116	
					0000G	30	BSBW	PRINT_OPR_MSG	
					10	CO	ADDL2	#16, SP	
					24	11	BRB	33\$	2074
					6E	F0	INSV	WRITE_RING, #25, #1, DEVCHAR	2079
					01	E0	BBS	#1, FLAGS, 32\$	2080
					02	E0	BBS	#2, FLAGS, 32\$	
					01	DD	PUSHL	#1	2081
					01	DD	PUSHL	#1	
					5E	DD	PUSHL	SP	
					CF	9F	PUSHAB	SET_MVL_OVERRIDE	
					04	FB	CALLS	#4, @SYSS\$CMKRNL	
					01	D0	MOVL	#1, R0	2083
					04	002A0	RET		
					50	D4	CLRL	R0	2084
					04	002A3	RET		

; Routine Size: 676 bytes, Routine Base: \$CODE\$ + 058B


```
: 1584      2085 1 ROUTINE SET_MVL_OVERRIDE ( VALUE ) : NOVALUE MVL_UCB =
: 1585      2086 1
: 1586      2087 1 ++
: 1587      2088 1
: 1588      2089 1 FUNCTIONAL DESCRIPTION:
: 1589      2090 1     this routine sets the MVL "can override" bit for this reel
: 1590      2091 1
: 1591      2092 1 CALLING SEQUENCE:
: 1592      2093 1     SET_MVL_OVERRIDE(ARG1)     KERNEL CALL!!!!
: 1593      2094 1
: 1594      2095 1 INPUT PARAMETERS:
: 1595      2096 1     ARG1 - the value to which the bit should be set ( passed by value )
: 1596      2097 1
: 1597      2098 1 IMPLICIT INPUTS:
: 1598      2099 1     NONE
: 1599      2100 1
: 1600      2101 1 OUTPUT PARAMETERS:
: 1601      2102 1     NONE
: 1602      2103 1
: 1603      2104 1 IMPLICIT OUTPUTS:
: 1604      2105 1     NONE
: 1605      2106 1
: 1606      2107 1 ROUTINE VALUE:
: 1607      2108 1     NONE
: 1608      2109 1
: 1609      2110 1 SIDE EFFECTS:
: 1610      2111 1     NONE
: 1611      2112 1
: 1612      2113 1 USER ERRORS:
: 1613      2114 1     NONE
: 1614      2115 1
: 1615      2116 1 --
: 1616      2117 1
: 1617      2118 2 BEGIN
: 1618      2119 2
: 1619      2120 2 EXTERNAL REGISTER
: 1620      2121 2     MVL_ENTRY = 9      : REF BBLOCK;
: 1621      2122 2
: 1622      2123 2 MVL_ENTRY [ MVL$V_OVERRIDE ] = .VALUE;
: 1623      2124 2
: 1624      2125 1 END;                                ! end of Routine SET_MVL_OVERRIDE
```

```
                                0000 00000 SET_MVL_OVERRIDE:
                                .WORD  Save nothing
07  A9                        01      02      04  AC  F0 00002  INSV  VALUE, #2, #1, 7(MVL_ENTRY) : 2085
                                04 00009  RET                                     : 2123
                                                                : 2125
```

; Routine Size: 10 bytes, Routine Base: \$CODE\$ + 082F

```
: 1625      2126 1
: 1626      2127 1 END
: 1627      2128 1
```

: 1628

2129 0 ELUDOM

PSECT SUMMARY					
Name	Bytes	Attributes			
SCODE\$	2105	NOVEC,NOWRT,	RD	EXE,NOSHR,	LCL, REL, CON,NOPI,ALIGN(2)
SLOCKEDD1\$	33	NOVEC, WRT,	RD	NOEXE,NOSHR,	LCL, REL, CON,NOPI,ALIGN(2)

Library Statistics					
File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	84	0	1000	00:01.8

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:MOUVOL/OBJ=OBJ\$:MOUVOL MSRC\$:MOUVOL/UPDATE=(ENHS:MOUVOL)

Size: 2071 code + 67 data bytes

Run Time: 00:42.8

Elapsed Time: 01:31.4

Lines/CPU Min: 2982

Lexemes/CPU-Min: 21224

Memory Used: 261 pages

Compilation Complete

0255 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

